

PATENT ABSTRACTS OF JAPAN

2

(11)Publication number : 2001-230770

(43)Date of publication of application : 24.08.2001

(51)Int.Cl.

H04L 9/10

G09C 1/00

(21)Application number : 2000-035898

(71)Applicant : TOSHIBA CORP

(22)Date of filing : 14.02.2000

(72)Inventor : HASHIMOTO MIKIO

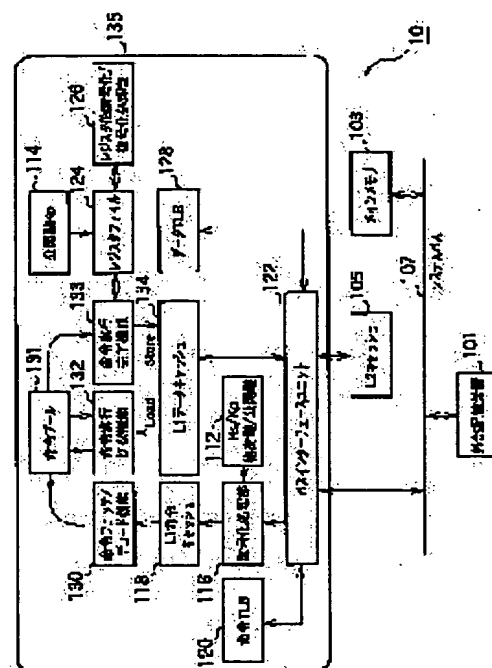
SAITO TAKESHI

TERAMOTO KEIICHI

(54) MICROPROCESSOR**(57)Abstract:**

PROBLEM TO BE SOLVED: To provide a microprocessor which can securely prevent the illegal rewriting of an operation, program.

SOLUTION: The microprocessor holds inside a specific secret key which cannot be read out and decodes contents are previously ciphered by a specific open key corresponding to the secret key. The bus interface unit of the microprocessor has an address conversion means converting a virtual address into a physical address and a memory read means reading contents corresponding to the physical address designated from a memory outside the microprocessor. Instruction TLB stores a table including an address conversion rule and one or above entries having ciphered attribute information in ranges designated by the virtual addresses. When ciphered attribute information shows that the range designated by the virtual address is ciphered, the corresponding contents are read from the outer memory through the memory reading means and it is decoded by a decoding processing section.

**LEGAL STATUS**

[Date of request for examination]

11.12.2002

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

MicroPatent® Worldwide PatSearch: Record 1 of 1

Family of JP2000035898 [How it Works](#)

MicroPatent® Family Lookup

Stage 2 Patent Family - "Extended"					Priorities and Applications			
CC	Document Number	KD	Publication Date		CC	Application or Priority Number	KD	Application or Priority Date
<input type="checkbox"/>	JP	2000035898	A	20000202	JP	20262598	A	19980717
<input type="checkbox"/>	JP	3189793	B2	20010716	JP	20262598	A	19980717

2 Publications found.
Information on the left side of the table relates to publication number, kind, and date; information on the right covers the corresponding application and priority data for each publication.
Legend: CC=Country Code KD=Kind (Publication kind can differ from application/priority kind.)

[Order Selected Documents](#)

Copyright © 2004, MicroPatent, LLC. The contents of this page are the property of MicroPatent, LLC including without limitation all text, html, asp, javascript and xml. All rights herein are reserved to the owner and this page cannot be reproduced without the express permission of the owner.

For further information, please contact:
[Technical Support](#) | [Billing](#) | [Sales](#) | [General Information](#)

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2000-35898

(P2000-35898A)

(43) 公開日 平成12年2月2日(2000.2.2)

(51) Int.Cl.⁷

G 0 6 F 11/22

識別記号

3 4 0

F I

G 0 6 F 11/22

テーマコード*(参考)

3 4 0 A 5 B 0 4 8

審査請求 有 請求項の数41 O L (全 47 頁)

(21) 出願番号

特願平10-202625

(22) 出願日

平成10年7月17日(1998.7.17)

(71) 出願人 000004237

日本電気株式会社

東京都港区芝五丁目7番1号

(72) 発明者 福島 裕

東京都港区芝五丁目7番1号 日本電気株式会社内

(74) 代理人 100086645

弁理士 岩佐 義幸

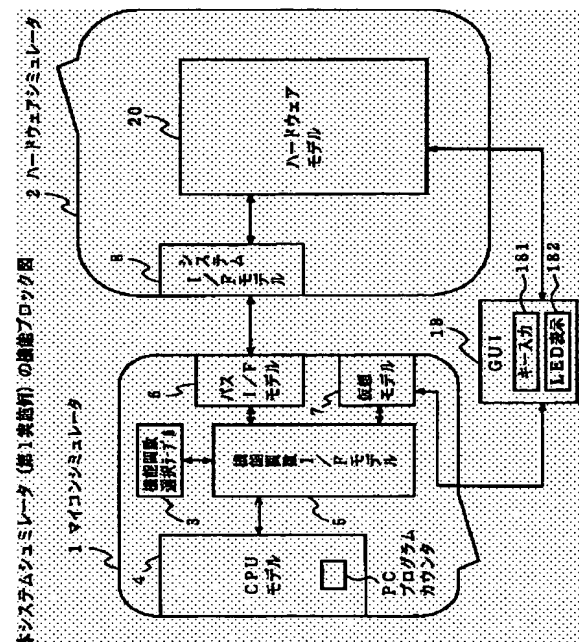
Fターム(参考) 5B048 AA12 AA20 DD14

(54) 【発明の名称】 システムシミュレータおよびシステムシミュレーション方法

(57) 【要約】

【課題】 マイコンを使用する電子機器のシステムシミュレーションにおいて、検証時間の短縮と検証内容の充実を図る。

【解決手段】 ハードウェアシミュレータ2は、電子機器の周辺回路をソフトウェアで模擬するハードウェアモデル20と、マイコンシミュレータ1との間で情報授受の整合をとるシステムI/Fモデル8とから成り、電子機器のハードウェアをソフトウェアで検証する。マイコンシミュレータ1のCPUモデル1はターゲットプログラムを検証する。このとき、ハードウェア機能関数I/Fモデル5を介して、仮想モデル7に登録されたハードウェアシミュレーションの結果を適時に利用する。機能関数I/Fモデル5は、CPUモデル42からの機能関数のうち機能関数選択テーブル3に登録されているものについては仮想モデル7を利用し、そうでないものについてはバスI/Fモデル6により情報授受の整合をとってハードウェアシミュレータ2を使用する。



【特許請求の範囲】

【請求項1】マイクロコンピュータを使用する電子機器のプログラムおよびハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレータにおいて、

前記プログラムに基づいて前記ハードウェアをソフトウェアで検証するハードウェアシミュレータと、
前記ハードウェアに係わる前記プログラム命令を前記ハードウェアと等価的にソフトウェアで処理する仮想モデルシミュレータと、
前記ハードウェアシミュレータまたは前記仮想モデルシミュレータの出力を適時に利用しながら前記プログラムをソフトウェアで検証するCPUモデルシミュレータとを具備することを特徴とするシステムシミュレータ。

【請求項2】前記CPUモデルシミュレータは、CPUインタフェースモデルを介して前記ハードウェアシミュレータまたは前記仮想モデルシミュレータのいずれかと接続されることを特徴とする請求項1記載のシステムシミュレータ。

【請求項3】前記CPUモデルシミュレータは、前記電子機器を構成する周辺回路に対する入出力命令を機能関数に変換して出力する手段を設けたことを特徴とする請求項1または請求項2に記載のシステムシミュレータ。

【請求項4】前記CPUインタフェースモデルは、前記ハードウェアシミュレータまたは前記仮想モデルシミュレータの一方を選択して前記CPUモデルシミュレータと接続する機能関数インタフェースモデルと、
前記仮想モデルシミュレータを選択するための条件を規定したシミュレータ選択テーブルとを有することを特徴とする請求項1～請求項3のいずれかに記載のシステムシミュレータ。

【請求項5】前記ハードウェアシミュレータは、前記電子機器を構成する周辺回路対応のサブモデルを含み前記ハードウェアの検証を実行するハードウェアモデルと、
該ハードウェアシミュレータと前記CPUモデルシミュレータとの間の情報授受の整合をとるシステムインタフェースモデルとを有することを特徴とする請求項1～請求項3のいずれかに記載のシステムシミュレータ。

【請求項6】前記CPUインタフェースモデルは、前記機能関数インタフェースモデルと前記システムインタフェースモデルとの間の整合をとるバスインタフェースモデルを有することを特徴とする請求項5に記載のシステムシミュレータ。

【請求項7】前記電子機器を構成する周辺回路対応のサブモデルは、前記周辺回路の接続情報を有していることを特徴とする請求項5に記載のシステムシミュレータ。

【請求項8】前記電子機器を構成する周辺回路対応のサブモデルは、前記プログラム・コード、または前記プログラムの処理結果、またはメモリモデルの記憶情報を保持する記憶手段を有していることを特徴とする請求項5

に記載のシステムシミュレータ。

【請求項9】前記仮想モデルシミュレータは、前記電子機器を構成する周辺回路に対応する命令の実行結果と等価な機能を実行する複数の仮想サブモデルを有することを特徴とする請求項1～請求項3のいずれかに記載のシステムシミュレータ。

【請求項10】前記仮想サブモデルは、前記プログラム・コード、または前記プログラムの処理結果、またはメモリモデルの記憶情報を保持する記憶手段を有することを特徴とする請求項9に記載のシステムシミュレータ。

【請求項11】前記仮想サブモデルは、複数の前記サブモデルに係わる前記プログラムを構成するサブルーチンの機能を代行する手段で構成されることを特徴とする請求項9または請求項10に記載のシステムシミュレータ。

【請求項12】前記仮想サブモデルは、前記サブモデルに係わる前記プログラムを前記サブモデルと等価な処理をする周辺装置で処理する手段で構成されることを特徴とする請求項9または請求項10に記載のシステムシミュレータ。

【請求項13】前記仮想モデルシミュレータは、シミュレーション結果に基づき、該当する前記サブモデルの記憶手段の記憶内容を更新することを特徴とする請求項9～請求項12のいずれかに記載のシステムシミュレータ。

【請求項14】前記ハードウェアモデルシミュレータは、シミュレーション結果に基づき、該当する前記仮想サブモデルの記憶手段の記憶内容を更新する手段を有することを特徴とする請求項9～請求項12のいずれかに記載のシステムシミュレータ。

【請求項15】前記ハードウェアシミュレータは、前記ハードウェアモデルからの割込み要求信号を前記CPUモデルに伝える割込インタフェースモデルを有し、前記ハードウェアの検証を前記プログラムの検証より優先して行うようにしたことを特徴とする請求項1～請求項14のいずれかに記載のシステムシミュレータ。

【請求項16】前記ハードウェアシミュレータ、前記ハードウェアモデルからのバス解放要求信号を前記CPUモデルに伝え、前記CPUモデルからのバス使用許可信号を前記ハードウェアモデルに伝えるバス要求インタフェースモデルを有し、前記プログラムの検証を中断するようにしたことを特徴とする請求項1～請求項15のいずれかに記載のシステムシミュレータ。

【請求項17】前記ハードウェアモデルと前記仮想モデルとの対応は、各モデルを構成するサブモデル間で1:N(Nは2以上の整数)であることを特徴とする請求項1～請求項16のいずれかに記載のシステムシミュレータ。

【請求項18】前記ハードウェアモデルまたは前記仮想モデルは、各モデルに所定のデータを与えたり、シミュ

レーション結果を表示するユーザーインタフェースと接続されたことを特徴とする請求項 1～請求項 17 のいずれかに記載のシステムシミュレータ。

【請求項 19】マイクロコンピュータとこれに接続される周辺回路とを有する電子機器のターゲットプログラムおよび周辺回路のハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレーション方法において、

前記ハードウェアの接続情報に基づいて周辺回路の処理をシミュレーションする第 1 ステップと、

前記ハードウェアの論理機能に基づいて周辺回路の処理を等価的にシミュレーションする第 2 ステップとを有し、

前記第 1 または第 2 ステップのいずれかを選択してマイクロコンピュータの入出力命令をシミュレーションする第 3 ステップとを備えたことを特徴とするシステムシミュレーション方法。

【請求項 20】マイクロコンピュータとこれに接続される周辺回路とを有する電子機器のターゲットプログラムおよび周辺回路のハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレーション方法において、

前記周辺回路の接続情報に基づいて周辺回路の処理をシミュレーションする第 1 ステップと、

前記周辺回路の機能と等価な機能を有する周辺装置を使用して周辺回路の処理結果を得る第 2 ステップとを有し、

前記第 1 または第 2 ステップのいずれかを選択してマイクロコンピュータの入出力命令をシミュレーションする第 3 ステップとを備えたことを特徴とするシステムシミュレーション方法。

【請求項 21】前記第 1 または第 2 ステップのいずれかを選択する前記第 3 ステップは、シミュレーション選択テーブルに基づいて行われ、前記シミュレーション選択テーブルの選択条件を満たすとき第 2 ステップを選択し、満たさないとき前記第 1 ステップを選択するようにしたことを特徴とする請求項 19 または請求項 20 に記載のシステムシミュレーション方法。

【請求項 22】前記第 1 または第 2 ステップのいずれかを選択する前記第 3 ステップは、シミュレーション選択テーブルに基づいて行われ、

前記シミュレーション選択テーブルにフラグが付加され、前記フラグが第 1 の状態であって前記選択テーブルの選択条件を満たすとき前記第 2 ステップを選択し、前記フラグが第 2 の状態であるとき前記第 1 ステップを選択することを特徴とする請求項 19 または請求項 20 に記載のシステムシミュレーション方法。

【請求項 23】前記第 1 及び第 2 ステップの処理結果を第 1 及び第 2 の記憶領域にそれぞれ記憶するステップを有し、

第 1 ステップを実行した後に、第 1 ステップの処理結果で第 2 の記憶領域を更新するステップと、

第 2 ステップを実行した後に、第 2 ステップの処理結果で第 1 の記憶領域を更新するステップとを備えたことを特徴とする請求項 19 または請求項 20 に記載のシステムシミュレーション方法。

【請求項 24】マイクロコンピュータとこれに接続される周辺回路とを備えた電子機器のターゲットプログラムおよび周辺回路のハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレーション方法において、

前記マイクロコンピュータが前記周辺回路との入出力命令を検出するステップと、

前記入出力命令に対応する機能関数を生成するステップと、

前記機能関数に応答し、前記ハードウェアの接続情報に基づいて前記周辺回路の処理をシミュレーションするハードウェアモデルシミュレーションと、

前記ハードウェアの論理機能に基づいて前記周辺回路の処理を等価的にシミュレーションする仮想モデルシミュレーションとを有し、

前記機能関数に基づき前記ハードウェアモデルシミュレーションまたは前記仮想モデルシミュレーションのいずれかを選択して前記入出力命令に対するシミュレーション結果を求めるステップを備えたことを特徴とするシステムシミュレーション方法。

【請求項 25】前記ハードウェアモデルシミュレーションまたは前記仮想モデルシミュレーションを選択する処理は、前記仮想モデルシミュレーションで処理可能な前記機能関数名と、前記機能関数の選択条件が登録されたシミュレーション選択テーブルにより行われることを特徴とする請求項 24 に記載のシステムシミュレーション方法。

【請求項 26】前記ハードウェアモデルシミュレーションまたは前記仮想モデルシミュレーションを選択する処理は、前記ハードウェアモデルシミュレーションで処理可能な前記機能関数名と、前記機能関数の選択条件と、フラグとが登録されたシミュレーション選択テーブルにより行われることを特徴とする請求項 24 に記載のシステムシミュレーション方法。

【請求項 27】前記仮想モデルシミュレーションへ前記ハードウェアモデルシミュレーションの初期状態と等価な状態を登録する処理はシステムシミュレーション開始前に行われていることを特徴とする請求項 24～請求項 26 のいずれかに記載のシステムシミュレーション方法。

【請求項 28】前記仮想モデルシミュレーションへ前記ハードウェアモデルの検証結果を登録する処理は前記システムシミュレーションの進行につれて更新されることを特徴とする請求項 24～請求項 27 のいずれかに記載

のシステムシミュレーション方法。

【請求項 2 9】前記更新は、前記仮想モデルに付属して設けられたテーブルまたはレジスタへの書き込みによって行うことを特徴とする請求項 2 8 に記載のシステムシミュレーション方法。

【請求項 3 0】前記システムシミュレーションの進行につれて前記仮想モデルシミュレーションの結果が登録され、その登録内容が前記ハードウェアモデルシミュレーションに反映されることを特徴とする請求項 2 4～請求項 2 8 のいずれかに記載のシステムシミュレーション方法。

【請求項 3 1】前記更新は、前記ハードウェアモデルに付属して設けられたテーブルまたはレジスタへの書き込みによって行うことを特徴とする請求項 3 0 に記載のシステムシミュレーション方法。

【請求項 3 2】前記周辺回路の検証は、前記マイクロコンピュータのシミュレーションにおける周辺回路処理サブルーチンとして行われることを特徴とする請求項 2 4 に記載のシステムシミュレーション方法。

【請求項 3 3】前記ハードウェアモデルシミュレーションの結果をもとに割り込み要求を検出するステップと、割り込みインタフェースモデルを介して前記マイクロコンピュータのモデルに割り込み要求信号を伝えるステップと、前記ターゲットプログラムに記述され、前記割り込み要求信号によって起動される割り込み処理によりハードウェアの検証を行うようにしたことを特徴とする請求項 2 4～請求項 3 2 のいずれかに記載のシステムシミュレーション方法。

【請求項 3 4】前記ハードウェアモデルシミュレーションの結果をもとにバス解放要求を検出するステップと、バス解放要求インタフェースモデルを介して前記マイクロコンピュータのモデルにバス解放要求信号を伝えるステップと、バス解放要求インタフェースモデルを介して前記ハードウェアモデルにバス使用許可信号を伝えるステップと、前記マイクロコンピュータモデルの検証を停止させて前記ハードウェアの検証を行うようにしたことを特徴とする請求項 2 4～請求項 3 2 のいずれかに記載のシステムシミュレーション方法。

【請求項 3 5】前記割り込み要求信号は前記周辺回路を構成するシリアル入出力インタフェース対応のサブモデルから出力され、また前記バス要求信号は前記周辺回路を構成する DMA 制御装置対応のサブモデルから出力されることを特徴とする請求項 3 4 に記載のシステムシミュレーション方法。

【請求項 3 6】前記周辺回路の検証は、前記割り込み要求信号発生の要因となる事象の発生により起動される前記ハードウェアモデルシミュレーションによって行われることを特徴とする請求項 3 4 または請求項 3 5 に記載のシステムシミュレーション方法。

【請求項 3 7】前記割り込み要求信号は前記シミュレーション装置の外部端子への通信データの受信により発生することを特徴とする請求項 3 6 に記載のシステムシミュレーション方法。

05 【請求項 3 8】前記ハードウェアの検証が、前記シミュレーション装置の外部端子で受信した通信データのデータメモリへの DMA 転送であることを特徴とする請求項 1 3～請求項 1 6 のいずれかに記載のシステムシミュレーション方法。

10 【請求項 3 9】マイクロコンピュータとこれに接続される周辺回路とを有する電子機器のターゲットプログラムおよび周辺回路のハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレーション方法において、

15 前記ターゲットプログラムはサブルーチン処理を有し、前記サブルーチン処理を該サブルーチンによって得られる結果と等価な結果が得られる仮想モデルシミュレーションで置き換えて処理することを特徴とするシステムシミュレーション方法。

20 【請求項 4 0】マイクロコンピュータを使用する電子機器のプログラムおよびハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレーション方法において、該システムシミュレーションの母体装置の読み出し専用メモリに、前記プログラムを書き込む手順

25 と、前記母体装置のメモリに、前記プログラムを検証するマイコンシミュレーションプログラム、前記プログラムに基づいて前記ハードウェアを検証するハードウェアモデルシミュレーションプログラム、CPU 情報、ハードウェア接続情報、仮想モデル情報、仮想モデル選択情報およびシミュレータ間インタフェース情報を書き込んで該母体装置をシミュレーション装置化する手順と、前記プログラムにしたがってシステムシミュレーションを実行する手順とから成り、前記仮想モデル情報に前記ハードウェアの検証結果を含ませる手順と、前記ハードウェアの検証のうち前記仮想モデル選択情報で指定された部分については前記マイコンシミュレーションプログラムにより代替して実行する手順とを有することを特徴とするシステムシミュレーション方法。

30 【請求項 4 1】請求項 1 9～請求項 4 0 のいずれかに記載のシステムシミュレーション方法を実行するプログラムを記録したコンピュータ読み込み可能なシステムシミュレーション記録媒体。

【発明の詳細な説明】

【0001】

45 【発明の属する技術分野】本発明は、マイクロコンピュータを搭載した電子機器のシステムシミュレータおよびシステムシミュレーション方法に関する。

【0002】

【従来の技術】電子機器のセットメーカーは、マイクロコンピュータ（以下、マイコンという）を搭載した電子

機器の開発において、新しい機能をセットに盛り込み他社との差別化を図ろうとすることが多い。このとき、セットメーカーは、開発期間を短縮するためには、新しいマイコンが完成する前に、電子機器に搭載するかどうかを検討する必要がある。

【0003】しかしながら、セットメーカーは、新製品のマイコンのサンプルを入手しなければハードウェアの開発を開始できない。開発できたとしても、マイコンと周辺回路とのインタフェース部で不具合が発見され、電子機器の回路構成を再設計しなければならない場合も多い。

【0004】また、マイコンと周辺回路を同一チップ上に組み込んだシステムLSIにおいては、試作したシステムLSIを評価してから、不具合を変更して再度試作したのでは開発期間が非常に延びてしまい、商機を逃しかねない。また、開発に要する費用も膨大になり、損失が大きい。

【0005】そこで、ソフトウェアの検証を行うマイコンシミュレータと、ハードウェアを検証するハードウェアシミュレータを合体させて（このような合体されたシミュレータをシステムシミュレータという）、シミュレーション装置で動作させ、ハードウェアのプロトタイプが出来ていない開発初期の段階からソフトウェアとハードウェアの双方を同時に、しかも短期間に検証と修正をすることが極めて重要になってきた。

【0006】マイコンシミュレータは、電子機器を構成するマイコンの機能をシミュレーションするCPUモデルシミュレータと、電子機器を機能させるプログラム（以下、ターゲットプログラムという）を記憶するプログラムメモリとから構成される。このターゲットプログラムをCPUモデルシミュレータ上で実行し、ターゲットプログラムが当初意図した順序で動作するか、あるいは、プログラムが暴走したり、止まってしまうことがないかなどを検証する。

【0007】ハードウェアシミュレータは、CPUの周辺に接続したメモリやI/Oポート、論理回路（以下、周辺回路と記す）などが期待通りに動作するか否かを検証する。この検証を実行するため、ハードウェアを構成するロジックやトランジスタの接続情報や、これらの駆動能力、寄生容量、遅延時間など各種のパラメータがハードウェアシミュレータに記憶される。

【0008】ハードウェアシミュレータは、シミュレーションするとき、ある周辺回路に所定の信号を入力し、その信号を接続情報をもとに周辺回路を構成するトランジスタに入力し、その出力を次のトランジスタに入力するといった処理を繰り返し、周辺回路からどのような信号がどのようなタイミングで出力されるかを演算によって求める。

【0009】この演算量は膨大であるため、1つの信号を1つの周辺回路に入力し、その結果を得るのに数10

～100msかかる。これに対し、CPUモデルシミュレータは、プログラムを1行実行するのに、数10μsで完了する。このように、ハードウェアシミュレータと組み合わせてソフトウェアのシミュレーションを実行する場合、ハードウェアシミュレータの演算速度でシミュレーション時間が決められてしまい、非常に長時間を要していた。

【0010】例えば、通常のキースキャンは、出力ポートからキーマトリクスに信号を出力し、キーマトリクスからの信号を入力ポートを通して入力することで、押下されたキーを探す処理である。出力ポートにはキーマトリクスの行数または列数に相当する回数だけ信号を出力する必要がある。入力ポートについては、キーマトリクスの行数または列数を入力ポートの端子数で除算した回数だけ信号を入力する必要がある。マイコンシミュレータとハードウェアシミュレータ間で、1回の入力または出力処理で、後述のように、4回の通信が必要である。例えば、キーマトリクスが4行×4列の場合、8回の入出力を実行しなければならない。従って、1回のキースキャンあたり、通信回数は $8 \times 4 = 32$ 回である。普通は、1回のキースキャンでキー押下を検出されることはなく、エンドレスで繰り返されるため、10回繰り返したとしても、320回の通信が発生する。

【0011】実際のキースキャン・プログラムは、チャタリング等による誤入力を防止するため、一度のキー押下を検出するだけではなく、一定時間経過後に再度キースキャンを実行し、同じキーが押され続けていることを確認し、押下されたキーを特定する。この処理をハードウェアシミュレータで行うと、前述のように数100回～数1000回の通信が必要であり、1つのキー押下を検出するだけで数分を要することになる。

【0012】このため、従来はプログラムとハードウェアの検証を別々に実行したり、周辺回路は実際に組み立て、これをマイコンシミュレータに接続して検証するなどしていた。

【0013】
【発明が解決しようとする課題】しかしながら、上述した従来のシミュレーション方法のうちの前者では、周辺回路に係わる命令のシミュレーションが十分に検証できず、電子機器に組み立ててから問題を発見することがあった。

【0014】また、後者は、周辺回路を組み立てる工数や部品代が必要であり、問題が見つかったときは、再度組み立て直さなければならない。また、周辺回路をICに組み込むとタイミングがずれることもあり、IC化したときに初めて問題に気付くこともあった。

【0015】本発明の目的は、電子機器を構成するハードウェアの検証を、電子機器に組み込むプログラムの検証と同時に実行することで、検証時間の短縮と、検証内容の充実を図ったシステムシミュレータおよびシステム

シミュレーション方法を提供することにある。

【0016】

【課題を解決するための手段】本発明のシステムシミュレータは、マイクロコンピュータを使用する電子機器のプログラムおよびハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレータにおいて、前記プログラムに基づいて前記ハードウェアをソフトウェアで検証するハードウェアシミュレータと、前記ハードウェアに係わる前記プログラム命令を前記ハードウェアと等価的にソフトウェアで処理する仮想モデルシミュレータと、前記ハードウェアシミュレータまたは前記仮想モデルシミュレータの出力を適時に利用しながら前記プログラムをソフトウェアで検証するCPUモデルシミュレータとを具備することを特徴とする。

【0017】また、本発明のシステムシミュレーション方法は、マイクロコンピュータとこれに接続される周辺回路とを有する電子機器のターゲットプログラムおよび周辺回路のハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレーション方法において、前記ハードウェアの接続情報に基づいて周辺回路の処理をシミュレーションする第1ステップと、前記ハードウェアの論理機能に基づいて周辺回路の処理を等価的にシミュレーションする第2ステップとを有し、前記第1または第2ステップのいずれかを選択してマイクロコンピュータの入出力命令をシミュレーションする第3ステップとを備えたことを特徴とする。

【0018】

【発明の実施の形態】次に、本発明の実施の形態について説明する。

【0019】本発明のシステムシミュレータは、マイクロコンピュータを使用する電子機器のプログラムおよびハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレータにおいて、前記プログラムに基づいて前記ハードウェアをソフトウェアで検証するハードウェアシミュレータと、該ハードウェアの検証結果を適時に利用しながら前記プログラムをソフトウェアで検証するマイコンシミュレータとから成ることを特徴とする。

【0020】また、本発明のシステムシミュレーション方法は、マイクロコンピュータを使用する電子機器のプログラムおよびハードウェアをシミュレーション装置上で一体的に検証するシステムシミュレーション方法において、該システムシミュレーションの母体装置の読み出し専用メモリに、前記プログラムを書き込む手順と、前記母体装置のメモリに、前記プログラムを検証するマイコンシミュレーションプログラム、前記プログラムに基づいて前記ハードウェアを検証するハードウェアシミュレーションプログラム、CPU情報、ハードウェア接続情報、仮想モデル情報、仮想モデル選択情報およびシミュレータ間インタフェース情報を書き込んで該母体装置

をシミュレーション装置化する手順と、前記プログラムにしたがってシステムシミュレーションを実行する手順とから成り、前記仮想モデル情報に前記ハードウェアの検証結果を含ませる手順と、前記ハードウェアの検証のうち前記仮想モデル選択情報で指定された部分については前記マイコンシミュレーションプログラムにより代替して実行する手順とを有することを特徴とする。

【0021】以下、本発明の実施例について図面を参照して詳細に説明する。

【0022】[第1実施例]

構成の概要

図1は本発明のシステムシミュレータの第1実施例を示す機能ブロック図である。本システムシミュレータは、マイコンシミュレータ1とハードウェアシミュレータ2とから成る。また、本シミュレータは、シミュレーションの母体装置となる、例えばワークステーションにロードされて実行されるが、図1におけるユーザインタフェース(GUI)18は、そのようなシミュレーション装置が提供するユーザインタフェースである。なお、これ以降、インタフェースはI/Fと記すこととする。

【0023】ハードウェアシミュレータ2は、シミュレーション対象となる電子機器のハードウェアをソフトウェアで実現したハードウェアモデル20を中核とし、本電子機器を制御するマイクロプロセッサ(以下CPUと記す)に接続される周辺回路の接続情報や各種パラメータなどが記憶されており、入力されたデータに対して、期待されるデータが、期待されるタイミングで出力されるか、すなわち、論理的な矛盾や動作タイミングの矛盾がないかを検証する。さらに、ハードウェアモデル20は、図3にその具体例を示すように、各周辺回路の機能記述モデルから構成される。

【0024】マイコンシミュレータ1は、検証するプログラムを1行単位に読み込んで、プログラムの記述や処理順序に問題ないかを検証し、CPUモデル4、機能関数I/Fモデル5、バスI/Fモデル6、仮想モデル7および機能関数選択テーブル3を有する。

【0025】CPUモデル4は、CPUの機能をソフトウェアで実現したもので、次の処理を実行する。

- ・プログラムカウンタPCで指定されるメモリアドレスのデータを読み込む。
- ・データ(オペレーションコードあるいはオペランド)を解釈する。
- ・解釈結果に基づきその命令を実行する。
- ・周辺回路に関係する命令の場合、機能関数を生成する。

- ・機能関数I/Fモデル5に機能関数を渡す。

【0026】機能関数I/Fモデル5は、ハードウェアシミュレータ2を使うか、仮想モデル7を使うか、その選択を判断し、選択された方に必要な情報を渡し、また選択された方から必要な情報を得る。得た結果はCPU

モデル4に渡す。なお、仮想モデル7を選択するかどうかの判断は、機能関数選択テーブル（具体例を図6に示す）3をもとに行う。機能関数選択テーブル3は、予めシステム開発者が作成し、本システムシミュレーション装置に組み込んであるものとする。

【0027】仮想モデル7は、本発明の特徴部分の一つであり、本実施例では、ハードウェアモデル20を構成する機能記述モデルのうち、予め設定されたものから成る。本実施例では、ある周辺回路のハードウェアシミュレーションの結果が他の周辺回路のシミュレーションに影響を及ぼさないものについてのみ仮想モデル7に登録できる。

【0028】バスI/Fモデル6は、ハードウェアシミュレータ2のシステムI/Fモデル8との間で情報の受け渡しを行う。すなわち、機能関数I/Fモデル5がCPUモデル4から受け取った機能関数をもとに、アドレス信号、データ信号、リード/ライト信号などをシステムI/Fモデル8を介してハードウェアモデル20に渡し、ハードウェアシミュレータ2でシミュレーションした結果をシステムI/Fモデル8を介して得る。

【0029】システムI/Fモデル8は、マイコンシミュレータ1のバスI/Fモデル6との間で情報の受け渡しを行う。バスI/Fモデル6を介して受け取ったアドレス信号、データ信号、リード/ライト信号などをもとに、ハードウェアシミュレータ2で演算を実行し、その結果をバスI/Fモデル6に返す。

【0030】バスI/Fモデル6およびシステムI/Fモデル8は、またマイコンシミュレータ1とハードウェアシミュレータ2との間のコード変換をも行う。

【0031】ユーザインタフェース18は、電子機器のボタンや表示を画面に表示させて、画面上のボタンをマウス等で押した結果を読み取って、その情報をシミュレーション対象のプログラムに入力したり、プログラムで処理した結果を画面に表示させたりする。このGUI18は、ハードウェアシミュレータ2だけでなく、仮想モデル7とも接続されており、双方と情報の授受を行う。

【0032】図2は、本システムシミュレーションを行うための母体となる装置、例えばワークステーションをシミュレーション装置化するのに必要なプログラムファイルと情報ファイルを示す図である。本ワークステーションEWSは、プロセッサS_CPU、読み出し専用メモリS_ROM、メモリS_RAM、割込制御装置S_INT、DMA装置S_DMA、表示装置と接続された表示制御装置S_DSP、キーボード、マウスと接続された入力部S_INP、通信端末と接続されたシリアル入出力装置S_SIOおよびハードディスクS_HDDなどから構成される。

【0033】ハードディスクS_HDDには、マイコンシミュレータ1が動作するためのマイコンシミュレーションプログラム、ハードウェアシミュレータ2が動作す

るためのハードウェアシミュレータプログラム、マイコンの型番やレジスタ等の情報であるCPU情報、ハードウェアモデル20についての前述のハードウェア接続情報、仮想モデル7に移植すべき周辺回路の機能記述モデルについての仮想モデル情報、マイコンシミュレーション時において仮想モデル7とハードウェアシミュレータ2のいずれを選択するか基準となる仮想モデル選択情報およびマイコンシミュレータ1とハードウェアシミュレータ2とで情報の授受を行うときに必要なコード変換表等のインタフェース情報であるシミュレータ間インタフェース情報の各ファイルを格納する。

【0034】これらの各ファイルはメモリS_RAMにロードされ、このメモリS_RAM上に図1のマイコンシミュレータ1およびハードウェアシミュレータ2が仮想的に形成される。すなわち、CPUモデル4はマイコンシミュレーションプログラムによって動作し、ハードウェアシミュレータ2はハードウェアシミュレーションプログラムによって動作する。また、CPUモデル4とハードウェアモデル20は、それぞれCPU情報とハードウェア情報によって構成される。さらに、仮想モデル7は仮想モデル情報にしたがって形成され、仮想モデル選択情報は機能関数選択テーブル3の内容となり、バスI/Fモデル6とシステムI/Fモデル8はシミュレータ間インタフェース情報によって構成される。

【0035】なお、マイコンシミュレーションの対象となるプログラムは、マイコンシミュレーションプログラムの一部としてメモリS_RAMにロードされる。また、図2の表示装置は、図1のGUI18のLED表示GUI182やキー入力GUI181を表示し、キーボードやマウスは、表示されたキー入力GUI181のボタンを選択することにより、所望の情報をシミュレータに供給するためのものである。ここで、LED表示GUI182は、表示装置の画面上に表示されたランプ画像であり、点灯したランプ画像と、消灯したランプ画像を有している。LED表示GUI182は、供給される信号の状態に応じて、点灯または消灯の画像を表示する。また、キー入力GUI181は、表示装置の画面上に表示されたスイッチ画像である。

【0036】なお、上述のワークステーションEWSの構成要素S_HDD、S_DMAやファイルは、代表的な一例を示すものであって、用途に応じて適宜追加あるいは削除しうるものである。

【0037】動作の概要
さて、以上のようにして構成された本シミュレータの動作の概要について説明する。ハードディスクS_HDDには、図2の各ファイルが格納されており、また読み出し専用メモリS_ROMにはシミュレーション対象のプログラム（ターゲットプログラム）が予め書き込まれているものとする。まず、ワークステーションEWSの電源を投入して起動後に、上述のマイコンシミュレーシ

ンプログラム、ハードウェアシミュレーションプログラム等をメモリS_RAMにロードしてシミュレーション装置化する。すると、図1に示したようなマイコンシミュレータ1およびハードウェアシミュレータ2が構成され、システムシミュレーションの環境が整備されたので、いよいよシミュレーションを実行することになる。

【0038】CPUモデル4は、シミュレーション対象プログラムをプログラムカウンタPCの値にしたがって順次に読み込む。CPUモデル4が読み出した命令がジャンプ命令BR、BNZ、減算命令SUBのように、周辺回路に関係がない場合、CPUモデル4内部でシミュレーションが実行される。一方、CPUモデル4が読み出した命令が周辺回路モデルに関係し、仮想モデル7に登録されていない場合、CPUモデル4は、生成した機能関数を、バスI/Fモデル6とシステムI/Fモデル8を介してハードウェアシミュレータ2に渡し、演算処理が実行され、その結果を得る。そして、ハードウェアシミュレータ2は、演算を実行し、その結果をマイコンシミュレータ1に返す。このような命令として、例えば、ある周辺回路のハードウェアシミュレーションの結果が他の周辺回路のシミュレーションに影響を及ぼすようなものを挙げることができる。

【0039】ここで、機能関数とは、マイコンシミュレータ1とハードウェアシミュレータ2とではコード体系が異なることが少なくないことから、これらのコード体系にとらわれることなく取り扱えるように設定された中間的なコードをいう。

【0040】一方、CPUモデル4が読み出した命令が周辺回路モデルに関係し、仮想モデル7に登録されている場合には、CPUモデル4は、生成された機能関数を、機能関数I/Fモデル5に渡し、仮想モデル7を参照し、所望の情報を得る。仮想モデル7は、プログラムを実行して、その結果を機能関数I/Fモデル5を介してCPUモデル4に返す。この場合には、マイコンシミュレータ1は、周辺回路モデルに関わる命令を実行するのに、ハードウェアシミュレータ2ではなく、仮想モデル7から命令の実行結果を受け取ることになる。

【0041】これにより、マイコンシミュレータ1とハードウェアシミュレータ2との間での情報のやりとりが低減でき、シミュレータ相互間の情報の受け渡し回数や待ち合わせ時間を削減できるのである。

【0042】〔構成の具体例〕次に、本実施例の具体例としてマイコンを使用した電子手帳のシステムシミュレーションをとり挙げ、より詳細に説明する。

【0043】図3は、この電子手帳の一部分を構成するハードウェアモデル20の詳細を示すブロック図であり、プログラムを記憶するプログラムメモリモデル10、データを記憶するデータメモリモデル11、電子手帳の時計を模擬する時計モデル17、電子手帳の押ボタン群への入力機能モデルであるキー入力モデル19およ

びGUI18への出力ポートを模擬するI/OポートCモデル16とで構成される。

【0044】キー入力モデル19は、電子手帳の押ボタン群を模擬した4行4列のキーマトリクスモデル13、デコーダモデル15への入力信号PB0、PB1を与えるI/OポートBモデル14、この入力信号を解釈して、キーマトリクスモデル13の4本の列線のうちのいずれか1本の列線を活性化する信号D0～D3を出力するデコーダモデル15、キーマトリクスモデル13で押下された接点を含む行線上の信号PA0～PA3を入力するI/OポートAモデル12から成る。

【0045】プログラムメモリモデル10、データメモリモデル11、I/OポートAモデル12、I/OポートBモデル14、デコーダモデル15、時計モデル17の各ハードウェアモデル（周辺回路対応のサブモデル）は、図2に示したハードウェアシミュレーションプログラムおよびハードウェア接続情報に基づいて、演算するための回路接続情報と各種パラメータをメモリS_RAMに記憶している。また、プログラムメモリモデル10は、ターゲットプログラムを記憶するテーブル（記憶領域）を有している。

【0046】また、プログラムメモリモデル10、データメモリモデル11、時計モデル17、I/OポートAモデル12、I/OポートBモデル14およびI/OポートCモデル16は、バスBUSを介してシステムI/Fモデル8と接続される。時計モデル17は、CPUモデル4に供給されるクロック信号HCLK（不図示）が供給され、これを計数した時刻情報を保持している。

【0047】デコーダモデル15からの信号D0～D3およびI/OポートCモデル16からの信号PC0は、GUI18へ導かれ、GUI18からの信号PA0～PA3がI/OポートAモデル12へのキーマトリクスモデル13側からの入力となる。すなわち、キーマトリクスモデル13はあくまで仮想的なものであるため、人の現実の操作はGUI18上で行われる。したがって、デコーダモデル15からの信号D0～D3は実際にはGUI18に入力し、表示装置にマトリクス状に配置された複数のキースイッチ（キー入力GUI181）が表示され、人がキーボードまたはマウスでGUI18上のキー入力GUI181のいずれかを選択することで、キーマトリクスモデル13からの信号PA0～PA3となってI/OポートAモデル12に入力するものである。

【0048】図4は、図3のハードウェアモデル20に対応する仮想モデル7の詳細な構成例を示し、プログラムメモリモデル10対応の仮想プログラムメモリモデルV71、時計モデル17対応の仮想時計モデルV72、I/OポートAモデルとI/OポートBモデル対応の仮想キー入力モデルV73、およびI/OポートCモデル対応の仮想LED表示モデルV74から成る。このような仮想モデル7の構成は、図2における仮想モデル情報

ファイルをワークステーションEWSにロードすることにより、実現されるものである。

【0049】本実施例では、前述のように、ある周辺回路のハードウェアシミュレーションの結果が、他の周辺回路のシミュレーションに影響を及ぼさないものについてのみ仮想モデル7に登録できる。このことから、時計モデル17とI/OポートCモデル16は、他の周辺回路とは独立して動作しており、CPUモデル4以外とデータの授受がないので仮想モデル7に登録できるのに対し、I/OポートBモデル14のシミュレーション結果は、I/OポートAモデル12に影響を及ぼすので、どちらか一方だけを仮想モデル7に登録することはできないのである。

【0050】仮想プログラムメモリモデルV71は、図5に例示するように、プログラムメモリモデル10に記憶されたターゲットプログラムの命令コードと同じ命令コードをテーブル（記憶手段）に予め記憶している。このテーブルには、仮想プログラムメモリモデルV71のアドレスに対応するシンボルと命令コードに相当する戻り値とが配置されている。

【0051】仮想時計モデルV72は、CPUモデル4に供給されるクロックVCLK（不図示）を計数し、時刻情報を保持している。クロックの計数処理は、予め登録されたプログラムに従って処理され、マイコンシミュレータ1が他の処理を実行中であっても常に実行されており、マイコンを実使用したときに相当する時刻を出力する。ハードウェアシミュレータ2は、シミュレーションに時間がかかるため、クロックを計数しきれないが、マイコンシミュレータ1は、ハードウェアシミュレータ2により処理速度が速いので、クロックを計数することができる。計数結果は現実の時刻とは違い、進み方が遅いが、マイコンを実際に動作させたときには現実の時刻になるように設定されているので、シミュレーション実行中に実際に使用したときに相当する時刻を知ることができる。

【0052】仮想キー入力モデルV73は、キー入力GUI181のサブルーチンが呼び出されたとき、仮想モデル7内に予め登録したキー入力サブルーチンを処理することで、GUI18上で押されたボタンに相当する文字情報、例えば「0」、「1」、「A」などを得ることができる。これに対して、ハードウェアモデル20のデコーダモデル15からは、バイナリ形式の[0, 1, 0, 0]をGUI18のキー入力GUI181に供給し、キー入力GUI181からは、バイナリ形式の入力信号PA0～PA3 [0, 0, 1, 0]をI/OポートA12に受給することで、CPUモデル4はターゲットプログラム内のキー入力サブルーチンを処理して、どのボタンが押されたかを判断する。

【0053】仮想LED表示モデルV74は、供給されたLMP**をもとに、GUI18のLED表示GUI

182に供給する信号に変換する処理を実行する。例えば、「LMP00」が呼び出されたとき、仮想LED表示モデルV74は「0」を出力し、「LMP01」のときには「1」を出力する。

05 【0054】GUI18のLED表示GUI182は、LEDが点灯した画像情報と、LEDが消灯した画像情報とを有しており、仮想LED表示モデルV74の指示に応じて、いずれか一方の画像情報を選択して表示装置に表示する。「0」が入力されたときには、LEDが消灯した画像がLED表示GUI182に表示され、
10 「1」が入力されたときには、LEDが点灯した画像がLED表示GUI182に表示される。

【0055】これに対して、CPUモデル4からハードウェアモデル20のI/OポートCモデル16にI/O
15 ポート出力命令が実行され、ポートPC0が「0」になれば、LEDが消灯した画像がLED表示GUI182に表示され、「1」になれば、LEDが点灯した画像がLED表示GUI182に表示される。

【0056】図6は、図1に示した機能関数選択テーブル3の内容例であり、機能関数I/Fモデル5が、仮想
20 モデル7を選択可能な機能関数とその判断条件を示す。CPUモデル4から渡された機能関数が次の場合に、機能関数I/Fモデル5は仮想モデル7を選択する。CPUモデル4から渡された機能関数が機能関数選択テーブル3に記述されていなかったり、あるいは、記述されて
25 いても、判断条件を満たさない場合には、機能関数I/Fモデル5はハードウェアモデル20を選択して検証する。

【0057】図6において、ADRSはCPUモデル4
30 がアクセスするアドレスを示す信号であり、I/Oはメモリを除く周辺回路の入出力命令であることを示す信号であり、RDは読み出しであることを示す信号であり、WRは書き込みであることを示す信号である。また、「READTIME」、「PORTC」、「KEYI
35 N」は、アドレスを示すラベルである。また、「D*」は任意のデータを意味し、「TIME」は時計モデル17または仮想時計モデルV72から読み出した時刻を意味する。また、「&」は、条件の論理積、即ち、「&」の前後に記述された条件が同時に満たされるとき、判定
40 条件が有効であることを意味する。

【0058】仮想モデル7を選択可能な機能関数とその判断条件は次の通りである。

【0059】■FETCH関数：仮想プログラムメモリ
45 モデルV71から命令を読み込む関数であり、引数は、アドレスADRSである。本例では、アドレスADRSが1000H以上で、2000H未満なら仮想モデル7を選択可能。

【0060】■IORD関数：メモリ以外の周辺回路を
50 読み出す関数で、引数はアドレスADRSと、I/Oと、RDである。本例では、アドレスADRSが「RE

ADTIME」であるときは、仮想モデル7を選択可能。

【0061】■IOWR関数：メモリ以外の周辺回路にデータを書き込む関数で、引数は、アドレスADRS、書き込みデータ「D*」、I/O、及びWRである。本例では、アドレスADRSが「PORTC」で、データは任意の値「D*」であれば選択可能。

【0062】■CALL関数：特定のサブルーチンを呼び出したときに生成される関数で、引数はサブルーチンの先頭アドレスADRSと、時刻TIMEである。本例では、アドレスADRSがラベルKEYINで、時刻が05H以上で15H以下の場合には、仮想モデル7を選択可能。

【0063】図6に示す機能関数選択テーブル3は一つの例であって、このテーブルに登録する機能関数は、シミュレーションする対象に応じて適宜書き換えることができる。また、引数の条件、およびその組み合わせも適宜書き換えることができる。例えば、CALL関数において、時刻TIMEの条件は必須ではない。また、I/O、RD、WRなども、機能関数名IORD、IOWRにより判断できる場合には無くてもよい。

【0064】図7は、機能関数I/Fモデル5における仮想関数変換テーブルと、その戻り値を示す。機能関数I/Fモデル5は、CPUモデル4からは、前述のような理由から機能関数を受け取る。しかし、CPUモデル4と仮想モデル7とは異なるソフトメーカーで開発されることがあり、CPUモデル4が使用している機能関数コードと、仮想モデル7が使用する仮想関数コードとは異なることが多い。そこで、CPUモデル4対応のコード（機能関数コード）と仮想モデル7対応のコード（仮想関数コード）とを本テーブルにより変換することにしたのである。なお、この仮想関数変換テーブルは、図2のCPU情報および仮想モデル情報により生成される。また、図7では、引数[メモリ、I/O]&[RD、WR]は省略している。

【0065】図8は、プログラムメモリモデル10および仮想プログラムメモリモデルV71のアクセス・アドレスと、そのアドレスに記憶された命令コード、および、その命令コードの意味を示すニモニック表示とを表にしたものである。

【0066】図8のターゲットプログラムリストは、電子手帳のボタン押下の正当性を検証するものであり、キー入力GUI181に対して、あるボタンが押されればLED表示GUI182を点灯表示し、他のボタンが押されれば、LED表示GUI182を消灯表示するというものである。

【0067】なお、プログラムリスト中、#はイメージ・データ指定、!は絶対アドレス指定、Hは16進数であることを示す。

【0068】図8に示すターゲット・プログラム・リス

トの1行目「1001」番地の「A201」は、キーセン
スを行うために、I/OポートBに「01H」出力す
る命令、2行目「1002」番地の「C501」は、I
/OポートAからデータを入力してレジスタAに格納す
る命令、3行目「1003」番地の「B204」は、レ
ジスタAから「01H」を減算する命令、4行目「10
04」番地の「7406」は、レジスタAが「0H」で
あれば「1006H」番地に分岐する命令、5行目「1
005」番地の「2002」は、無条件で「1002
H」番地に分岐する命令、6行目「1006」番地の
「5AB5」は、時計モデルからデータを読み込んでレ
ジスタAに格納する命令、7行目「1007」番地の
「B310」は、レジスタAから「10H」を減算する
命令、8行目「1008」番地の「740A」は、レジ
スタAが「0H」であれば「100AH」番地に分岐す
る命令、9行目「2006」は、無条件で「1006
H」番地に分岐する命令、10行目「100A」番地の
「C701」は、I/OポートCに「01H」を出力す
る命令、11行目「100B」番地の「5A03」は、
サブルーチン「KEYIN」を呼び出す命令、12行目
「100C」番地の「B2EC」は、レジスタAから
「ECH」を減算する命令、13行目「100D」番地
の「740F」は、レジスタAが「0H」であれば「1
00FH」番地に分岐する命令、14行目「100E」
番地の「200B」は、無条件で「100BH」番地に
分岐する命令、15行目「100F」番地の「C70
0」は、I/OポートCに「00H」を出力する命令で
ある。

【0069】また、17行目の「KEYIN」はラベル
を表し、24行目「301F」番地の「RET」はサブ
ルーチンが呼び出された番地に戻ることを表す。また、
17行目から24行目までのリストは、キー入力のサブ
ルーチンが記述されており、キーマトリクスモデル13
をキースキャンして、押されたキーに対応付けられた文
字コードを戻すものとする。

【0070】図9は、CPUモデル4のシミュレーショ
ンプログラムのフローチャートを示す。

【0071】図9において、ステップS101で、CP
Uモデル4は、プログラムカウンタPCの値を取得す
る。CPUモデル4は、プログラムカウンタPCの値を
取得後、次の取得に備えてプログラムカウンタPCの値
を+1だけインクリメントする。

【0072】ステップS102で、CPUモデル4は、
プログラムカウンタPCで示されるアドレスをアクセス
して、プログラムメモリモデルから命令コードを読み出
すため、機能関数FETCH(PC)を生成する。ここ
で、PCはプログラムカウンタPCの値である。

【0073】ステップS103で、CPUモデル4は、
メモリを含む周辺回路をアクセスして所定の処理を実行
する周辺回路処理サブルーチンを呼び出す。ここでは、

機能関数F E T C H (P C) を機能関数 I / F モデル 5 に渡して、プログラムメモリモデルから命令コードを読み出す処理を実行する。

【0074】ステップS104で、CPUモデル4は、周辺回路処理サブルーチンにより得られた命令コードを命令レジスタに格納する。

【0075】ステップS105で、CPUモデル4は、命令レジスタに格納された命令コードを解読する。

【0076】ステップS105で、CPUモデル4は、解読された命令コードが、ロード命令またはストア命令であって、周辺回路との入出力(読み書き)が生ずるかどうかを判断する。周辺回路との入出力が生ずる場合には「Y」へ進み、ステップS107の処理を実行する。周辺回路との入出力が生じない場合には「N」へ進み、ステップS109の処理を実行する。

【0077】ステップS107で、CPUモデル4は、解読された命令コードに基づき周辺回路に対してデータを入力または出力する処理を実行するため、ロード命令またはストア命令に対応する機能関数を生成する。

【0078】ステップS108で、CPUモデル4は、メモリを含む周辺回路をアクセスして所定の処理を実行する周辺回路処理サブルーチンを呼び出す。このとき、CPUモデル4は、機能関数 I / F モデル 5 に機能関数を渡すことにより、周辺回路に対するロード命令またはストア命令が処理される。このサブルーチンは、ステップS103で呼び出されるサブルーチンと別であってもよいが、ここでは同一のサブルーチンを呼び出すものとして説明する。

【0079】ステップS109で、CPUモデル4は、CPUモデル4内で処理される分岐命令や加減算命令などを実行したり、周辺回路処理サブルーチンにより得られたデータを所定のレジスタに格納したりするなどの処理を実行する。

【0080】ステップS110で、CPUモデル4は、実行した命令が「B R E A K」や「S T O P」であれば「Y」へ進み、シミュレーションの処理を一旦停止したり終了したりする。そうでなければ「N」へ進み、ステップS101へ戻り、ステップS101からS109の処理を繰り返す。

【0081】図10は、図9における周辺回路処理サブルーチンS103、S108の詳細フローチャートを示す。

【0082】図10において、ステップS201で、機能関数 I / F モデル 5 は、CPUモデル4から機能関数を受け取り、この機能関数が機能関数選択テーブル3の条件を満たすかどうかを判断する。即ち、仮想モデル7を使って検証するか、ハードウェアシミュレータ2を使ってシミュレーションするかを判断する。機能関数選択テーブル3の判断条件を満たし、仮想モデル7を使う場合には、「Y」へ進み、ステップS202を実行する。機

能関数選択テーブル3の判断条件を満たし、仮想モデル7を使う場合には、「Y」へ進み、ステップS202を実行する。そうでなければ、「N」へ進み、ステップS203を処理する。このとき、機能関数 I / F モデル 5 は、仮想モデル7、または、バス I / F モデル 6 に機能関数を渡す。

【0083】ステップS202で、仮想モデル7は、機能関数 I / F モデル 5 より機能関数を受け取り、後述の仮想モデル・シミュレーションを実行する。実行後、サブルーチンが呼び出されたステップS103または、S108の次のステップへ戻る。

【0084】ステップS203で、バス I / F モデル 6 は、機能関数 I / F モデル 5 より受け取った機能関数を、ハードウェアシミュレータ2のシミュレーションに必要な端子信号コード(CS、ADRS、DATAなど)に変換し、所定のマシンサイクルでシステム I / F モデル8に渡す。

【0085】ステップS210で、バス I / F モデル 6 は、端子信号コードをシステム I / F モデル8に渡すとき、端子信号コード名称の変換を行ったり、信号授受のタイミングを調整したりする。一般に、シミュレーションを実現するソフトウェアは、複数のベンダから供給されることが多く、シミュレータ間のコード体系の違いを吸収したり、信号授受のタイミングの違いを調整する必要がある。このような問題がない場合には、ユーザはステップ210を省略してもよい。

【0086】ステップ211で、システム I / F モデル 8 は、バス I / F モデル 6 から端子信号コードを受け取り、これをハードウェアシミュレータ2に渡す。

【0087】ステップ209で、ハードウェアシミュレータ2は、受け取った端子信号コードをもとにシミュレーションを実行し、その結果をシステム I / F モデル8に渡す。

【0088】ステップ211で、システム I / F モデル 8 は、ハードウェアシミュレータ2からシミュレーション結果を受け取り、バス I / F モデル 6 にこれを渡す。

【0089】ステップS210で、バス I / F モデル 6 は、システム I / F モデル 8 からシミュレーション結果コードを受け取るとき、結果コード名称の変換を行ったり、信号授受のタイミングを調整したりする。

【0090】ステップS203で、バス I / F モデル 6 は、システム I / F モデル 8 から受け取った結果コードを、機能関数 I / F モデル 5 に渡し、機能関数 I / F モデル 5 はこれをCPUモデル4に渡す。

【0091】以上で、一連の周辺回路処理サブルーチンを終了し、サブルーチンが呼び出されたステップS103または、S108の次のステップへ戻る。

【0092】次に、バス I / F モデルシミュレーションS203とシステム I / F モデルシミュレーション内の詳細な動作を説明する。

【0093】ステップS204で、バスI/Fモデル6は、機能関数I/Fモデル5から受け取った機能関数を、ハードウェアシミュレータ2のシミュレーションに必要な端子信号コード(CS、ADRS、DATAなど)に変換し、所定のマシンサイクルでシステムI/Fモデル8に渡す準備を行う。

【0094】図12に、バスI/Fモデル6がハードウェアシミュレータ2と授受する代表的な端子信号コードと、その信号変化を表すタイミングチャートを示す。

【0095】図12において、マシンサイクルは、マイコンのクロックに相当し、ここでは、4マシンサイクルで1つのロード命令またはストア命令に基づく端子信号コードが授受されるものとする。/CSはチップ選択端子信号、ADRSはアドレス端子信号、/RDはリード端子信号、/WRはライト端子信号、DATAはデータ端子信号である。なお、「/」は信号がローレベル“0”のとき有効であることを意味する。

【0096】マシンサイクルM1の時点では、チップ選択端子信号/CSと、アドレス端子信号ADRSとが出力され、マシンサイクルM2では、リード端子信号/RDまたはライト端子信号/WRと、データ端子信号DATAとが出力される。マシンサイクルM3では、リード端子信号/RDまたはライト端子信号/WRと、データ端子信号DATAとが非活性化された信号が出力される。マシンサイクルM4では、チップ選択端子信号/CSと、アドレス端子信号ADRSとが活性化された信号が出力され、端子信号の授受が終了する。

【0097】なお、本実施例では、システムI/Fモデル8は、第1～第4マシンサイクルの端子信号コードの変化を一括して受け取り、ハードウェアモデルシミュレーションS209を行った結果をバスI/Fモデル6に返すものとして説明する。従って、読み出しデータDATAは第2マシンサイクルでは出力されず、シミュレーション結果を返すタイミングで出力される。別の方法として、ハードウェアシミュレータ2は、1マシンサイクル分の端子信号コードを受け取る毎にハードウェアモデルシミュレーションS209を実行することもできる。この方法であれば、システムI/Fモデル8は、第2マシンサイクルで読み出しデータDATAを出力することができる。

【0098】ステップS205で、バスI/Fモデル6は、ステップS210を介してシステムI/Fモデル8に第1マシンサイクルから順番に第4マシンサイクルまで端子信号コードを出力する。

【0099】ステップS213で、システムI/Fモデル8は、複数の端子信号コードの変化を判断し、変化がなければ、「N」に進み、ステップS213に戻り変化を待つ。複数の端子信号コードのうちいずれか1つが変化すれば、入力データの準備ができたと判断し、「Y」へ進み、ステップS214に移る。

【0100】ステップS214で、システムI/Fモデル8は、バスI/Fモデル6から1マシンサイクル分の端子信号コードを受け取る。

【0101】ステップS215で、システムI/Fモデル8は、バスI/Fモデル6から4マシンサイクル分の端子信号コードの受領が完了したかを判断する。未完了であれば、「N」へ進み、ステップS212に移り、完了すれば、「Y」へ進み、ステップS209の処理に移る。

【0102】ステップS209で、ハードウェアモデル2.0は、受領した端子信号コードをもとにハードウェアモデルシミュレーションを実行し、結果をシステムI/Fモデル8に返す。

【0103】ステップS212で、システムI/Fモデル8は、ステップS210を介してバスI/Fモデル6にシミュレーション結果または端子信号コード受領結果を返す。システムI/Fモデル8は、端子信号コードを正確に受領すると受信確認信号ACKを返し、正確に受領できないときは再送信を要求するため受信不能信号NACKを返し、ロード命令の実行後でシミュレーション結果があるときは出力の端子信号コードを返し、ストア命令の実行後でシミュレーション結果がないときはシミュレーション完了信号DONEを返す。

【0104】ステップS206で、バスI/Fモデル6は、システムI/Fモデル8から前記端子信号の変化があったか否かを判断する。変化がなければ、「N」に進み、ステップS206に戻り変化を待つ。複数の端子信号コードのうちいずれか1つが変化すれば、入力データの準備ができたと判断し、「Y」へ進み、ステップS207に移る。

【0105】ステップS207で、バスI/Fモデル6は、システムI/Fモデル8から端子信号コードまたはシミュレーション結果を受け取る。

【0106】ステップS208で、バスI/Fモデル6は、システムI/Fモデル8から受領したコードをもとに、4マシンサイクル分の端子信号コードの送信が完了したか、あるいは、シミュレーションが完了したか否かを判断する。未完了あるいはNACKであれば、「N」へ進み、ステップS205に移る。完了すれば、「Y」へ進み、一連の周辺回路処理サブルーチンを終了し、サブルーチンが呼び出されたステップS103または、S108の次のステップへ戻る。

【0107】図11は、図10における仮想モデルシミュレーションS202の詳細フローチャートを示す。

【0108】図11において、ステップS301で、仮想モデル7は、機能関数I/Fモデル5から機能関数を受け取ると、機能関数を仮想モデル7のコード体系にある仮想関数に変換する(図7)。例えば、FETCH(m)はMEMmに、IORD(READTIME)はR_TMに、IOWR(PORTC, n)はLMPn

に、CALL (KETIN) はK_INにそれぞれ変換される。なお、コード体系が同じであれば、本ステップは省略してもよい。

【0109】ステップS302で、仮想モデル7は、仮想関数がFETCH関数MEMmであるかを判断し、MEMmであれば、「Y」へ進み、ステップS307へ移り、MEMmでなければ、「N」へ進み、ステップS303へ移る。

【0110】ステップS303で、仮想モデル7は、仮想関数が時刻読出関数R_TMであるかを判断し、R_TMであれば、「Y」へ進み、ステップS308へ移り、R_TMでなければ、「N」へ進み、ステップS304へ移る。

【0111】ステップS304で、仮想モデル7は、仮想関数がキー入力関数K_INであるかを判断し、K_INであれば、「Y」へ進み、ステップS309へ移り、K_INでなければ、「N」へ進み、ステップS305へ移る。

【0112】ステップS305で、仮想モデル7は、仮想関数がLED点灯/消灯関数LMPであるかを判断し、LMPであれば、「Y」へ進み、ステップS310へ移り、LMPでなければ、「N」へ進み、ステップS306へ移る。

【0113】ステップS306で、仮想モデル7は、受領した仮想関数が登録されていない関数であったり、パラメータが違っていたりすると、表示装置にエラーメッセージを表示する。

【0114】ステップS307で、仮想モデル7内の仮想プログラムメモリモデルV71 (図4) は、予め登録されたターゲットプログラムのテーブル (図5) を参照し、シンボルMEMmに対応する戻り値を読み出し、機能関数I/Fモデル5に返し、処理を終了する。

【0115】ステップS308で、仮想モデル7内の仮想時計モデルV72 (図4) は、読み出し要求のあった時点の時刻を戻り値として機能関数I/Fモデル5に返し、処理を終了する。なお、仮想時計モデルV72は、予め登録された時計処理プログラムを有しており、このプログラムに従ってクロック信号VCLKを常に計数し、時刻情報を生成する。

【0116】ステップS309で、仮想モデル7内の仮想キー入力モデルV73 (図4) は、キー入力GUI181に対してキー入力を要求し、押下されたキーの文字情報を取得し、その文字情報を戻り値として機能関数I/Fモデル5に返し、処理を終了する。なお、本実施例では、キー入力GUI181により文字情報を取得する例を示すが、図2に示すキーボードで押されたキーを検出し、入力部S_INPを介して押下されたキーの文字情報を取得するようにしてもよい。

【0117】ステップS310で、仮想モデル7内の仮想LED表示モデルV74 (図4) は、受領した機能関

数I/Fモデル5をもとに、LED表示GUI182にランプの点灯画像、または消灯画像を表示させる。例えば、仮想関数がLMP00であれば、LED表示GUI182に「0」を出力して消灯表示にし、LMP01であれば、LED表示GUI182に「1」を出力して点灯表示にする。処理が完了すると、機能関数I/Fモデル5に完了信号DONEを返し、処理を終了する。

【0118】図11に示す上述の処理が終了すると、図10のステップS202の次のステップ「戻る」に戻る。

【0119】[動作の詳細] 次に、図8に示したターゲットプログラムのリストに基づいて、本電子手帳のシステムシミュレーションの動作につき、図5のプログラムメモリテーブル内のテーブル、図9～図11のフローチャートおよび図12のタイミングチャートを参照しながら、1つのプログラムステップごとに詳述する。

【0120】《ROMアドレス1001Hの処理》図9のステップS101で、CPUモデル4は、プログラムカウンタPCの値「1001H」を取得する。

【0121】ステップS102で、CPUモデル4は、機能関数FETCH (1001H) を生成し、機能関数モデル5に渡す。

【0122】ステップS103で、これにより、周辺回路処理サブルーチンが呼出される。

【0123】図10のステップS201で、機能関数I/Fモデル5は、機能関数選択テーブル3 (図6) に関数が登録されていて、且つその引数の条件を満たしているか、チェックを行う。今の場合、FETCH関数は登録されており、アドレスの条件を満たしているので、機能関数を仮想モデル7渡し、「Y」に進む。

【0124】ステップS202で、仮想モデル7は、仮想モデルシミュレーションを実行する。

【0125】図11のステップS301で、仮想モデル7は、機能関数I/Fモデル5より受領した機能関数FETCH (1001H) を、仮想関数変換テーブル (図7) をもとに仮想関数MEM1001に変換する。

【0126】ステップS302で、仮想モデル7は、変換した仮想関数がプログラムメモリを読み出すための関数 (MEM関数) かを判断する。今の場合、MEM関数なので、「Y」へ進む。

【0127】ステップS307で、仮想モデル7は、仮想プログラムメモリモデルV71 (図4) 内のテーブル (図5) を参照し、ラベル「MEM1001」に対応する戻り値「A201」を取得し、機能関数I/Fモデル5にこれを返す。

【0128】図11と図10の処理を終了し、図9に戻る。

【0129】ステップS104で、機能関数I/Fモデル5は、CPUモデル4に戻り値「A201」を渡すとともに、CPUモデル4は、戻り値「A201」を読み

込む。

【0130】ステップS105で、CPUモデル4は、戻り値「A201」を解析し、「MOV PORTB, #01H」であると解釈する。

【0131】ステップS106で、CPUモデル4は、05 解釈した命令が周辺回路の読み書きに関するものかを判断する。今の場合、I/OポートBモデルへのストア命令であるので、「Y」に進む。

【0132】ステップS107で、CPUモデル4は、010 解釈した命令に対応する機能関数IOWR (PORTB, #01H) を生成し、この機能関数を機能関数I/Fモデル5に渡す。

【0133】ステップS108で、CPUモデル4は、周辺回路処理サブルーチンの呼出しを実行する。

【0134】図10のステップS201で、機能関数I/Fモデル5は、仮想モデル7に引数の条件を満たす関数IOWR (PORTB, #01H) が登録されているかどうかのチェックを行う。この関数は図6の機能関数選択テーブル3に登録されていないので、機能関数I/Fモデル5は、機能関数IOWR (PORTB, #01H) をバスI/Fモデル6に渡し、「N」に進む。

【0135】ステップS203で、バスI/Fモデル6は、バスI/Fモデルシミュレーションを開始する。

【0136】ステップS204で、バスI/Fモデル6は、受領した機能関数IOWR (PORTB, #01H) をもとに、端子信号コード (図12) に変換し、システムI/Fモデル8へ出力する準備を行う。ここで、バスI/Fモデル6は、第1～第4マシンサイクルM1～M4に対応する/CS (チップ選択端子信号), AD RS (アドレス端子信号), /RD (リード端子信号), /WR (ライト端子信号) の各信号を生成する。なお、「/」は信号がローレベル“0”のとき有効であることを意味する。

【0137】ステップS205で、バスI/Fモデル6は、第1マシンサイクルM1の端子信号を出力する。すなわち、図12に示すように、マシンサイクルM1の時点での各端子信号は、次の状態である (図12)。

【0138】・I/OポートBモデル14用のチップ選択端子信号/CS=0

・アドレス端子信号ADRS=I/OポートBモデル14のアドレス

・データ端子信号DATA=不定

・リード端子信号/RD=1

・ライト端子信号/WR=1

ステップS210で、バスI/Fモデル6は、マイコンシミュレータ1内のCPUモデル4用の端子信号コード (/CS, ADRS等) から、ハードウェアシミュレータ2用の端子信号コード (不図示) に変換し、システムI/Fモデル8に各端子信号を渡す。これにより、ハードウェアシミュレータ2側の処理に移行する。

【0139】ステップS213で、システムI/Fモデル8は前記端子信号のいずれか1つの信号が変化したかを常に監視しており、端子信号のいずれかが変化したことを検出すると、入力データの準備が完了したと判断し、「Y」へ進む。

【0140】ステップS214で、システムI/Fモデル8は、/CS, ADRS, /RD, /WR, DATAの各端子信号を受け取る。

【0141】ステップS215で、システムI/Fモデル8は、所定のマシンサイクル (4回) に達しているかどうかを判断する。この時点では4回に達していないので、「N」へ進む。

【0142】ステップS212で、システムI/Fモデル8は、マシンサイクルM1の情報を正確に受信したことをバスI/Fモデル6に知らせるために、「ACK」 (受信確認) をバスI/Fモデル6に渡す。正確に受信できなかった場合には、「NACK」 (受信不能) を返し、バスI/Fモデル6に再送を促す。

【0143】ステップS210で、システムI/Fモデル8は、「ACK」データを変換して、バスI/Fモデル6に渡す。これにより、再びマイコンシミュレータ1側の処理に移る。

【0144】ステップS206で、バスI/Fモデル6は、端子信号のいずれかが変化したことを検出すると、025 入力データの準備が完了したと判断し、「Y」へ進む。

【0145】ステップS207で、バスI/Fモデル6は、「ACK」を受け取る。

【0146】ステップS208で、バスI/Fモデル6は、「ACK」受信なので、前端子信号は送信完了と判断する。現マシンサイクルは「M1」なので、通信は完了していないと判断し、「N」へ進む。

【0147】ステップS205で、バスI/Fモデル6はシステムI/Fモデル8にマシンサイクルM2の状態を渡す。マシンサイクルM2の時点での各端子信号は、35 次の状態である (図12)。

【0148】・I/OポートBモデル14用のチップ選択端子信号/CS=0

・アドレス端子信号ADRS=I/OポートBのアドレス

40 ・データ端子信号DATA=#01H

・リード端子信号/RD=1

・ライト端子信号/WR=0 (書き込みデータが有効であることを示す)

以下、ステップS210→S213→S214→S215→S212→S210→S206→S207→S208→S205の処理を、ステップS215においてマシンサイクルM4終了を検出するまで繰り返す。

【0149】ステップS215で、システムI/Fモデル8は、規定のマシンサイクルが完了したことを確認すると、受信した端子信号コードをハードウェアモデル250

0に渡し、「Y」へ進む。

【0150】ステップS209で、ハードウェアモデルシミュレーションを実行する。すなわち、図1のI/OポートBモデル14は、接続情報に基づき演算を行い、出力ポートに出力する値として(PB3, PB2, PB1, PB0) = (0, 0, 0, 1)を得る。このうち(PB1, PB0) = (0, 1)をデコーダモデル15に渡し、デコーダモデル15は、これをデコードして(D3, D2, D1, D0) = (0, 0, 1, 0)をキーマトリクスモデル13に出力するとともに、キー入力GUI181に渡す。

【0151】ステップS212で、ハードウェアモデル20内のI/OポートBモデル14は、上記シミュレーションの完了を知らせるため、シミュレーション完了信号「DONE」をシステムI/Fモデル8へ渡す。なお、シミュレーションでエラーが発生した場合には、シミュレーションエラー信号「ERR」を返す。

【0152】ステップS210で、システムI/Fモデル8は「DONE」をデータ変換する。

【0153】ステップS206で、バスI/Fモデル6は、端子信号の変化を検出し、「Y」へ進む。

【0154】ステップS207で、バスI/Fモデル6は、「DONE」を受信する。

【0155】ステップS208で、バスI/Fモデル6は、「DONE」受信により、バスI/Fが完了したと判断し、機能関数I/Fモデル5を介してCPUモデル4に、「DONE」を返し、「Y」に進む。

【0156】以上により、図9に戻り、ステップS108が終了する。

【0157】図9のステップS109で、CPUモデル4は、機能関数I/Fモデル5より「DONE」を受領することで、ストア命令の実行が終了したと検知する。

【0158】ステップS110で、停止要求ではないので、「N」へ進み、ステップS101に移る。

【0159】《ROMアドレス1002Hの処理》ステップS101で、CPUモデル4は、プログラムカウンタの値「1002H」を取得する。以下、1行目と同様に、ステップS102→S103→201→S202→S301→S302→S307と進む。

【0160】図9のステップS104で、CPUモデル4は、戻り値「C501H」(MOV A, PORT A)を読み込む。

【0161】ステップS105で、命令を解析し、ステップS106で、「Y」へ進む。

【0162】ステップS107で、CPUモデル4は、解説した命令に対応する機能関数IORD(PORT A, I/O, RD)を生成し、機能関数I/Fモデル5に渡す。

【0163】ステップS108で、周辺回路処理サブルーチン呼出し図10のステップS201で、この機能関

数IORD(PORT A)は図6の機能関数選択テーブル3に登録されていないので「N」へ進む。

【0164】ステップS204で、バスI/Fモデル6は、各端子信号を出力する準備をする。

05 【0165】ステップS205～S208で、前記動作を繰り返す。

【0166】ステップS209のシミュレーションの途中で、シミュレータのオペレータが、図2に示す入力手段により画面に表示されたキー入力GUI182のキーボードの右から2列目の上から3番目(図3)をクリックすることで、キー入力GUI182はその情報をI/OポートAモデル12の入力端子PA3～PA0に伝える。この結果、I/OポートAモデル12は、(PA3, PA2, PA1, PA0) = (0, 1, 0, 0)を得る。

10 【0167】ステップS212で、I/OポートAモデル12は、シミュレーション結果として「#04H」をシステムI/Fモデル8に渡す。

15 【0168】ステップS210で、システムI/Fモデル8は「#04H」をデータ変換する。

【0169】ステップS206で、「Y」へ進む。

【0170】ステップS207で、バスI/Fモデル6は、「#04H」を受領し、機能関数I/Fモデル5にこれを渡す。

20 【0171】ステップS208で、機能関数I/Fモデル5は、「#04H」をCPUモデル4に渡し、サブルーチン処理を終了し、図9のステップ108に戻る。

【0172】図9のステップS109で、CPUモデル4は、「#04H」をレジスタAに格納する。

25 【0173】ステップS110で、「N」に進み、ステップS101に戻る。

【0174】《ROMアドレス1003Hの処理》上記と同様、ステップS101→S102→S103→S201→S202→S301→S302→S307→S104→S105と進む。

30 【0175】ステップS106で、解説結果「SUB A #04H」は、周辺回路読み書きに関係ないので、機能関数の呼び出しは行われず、「N」へ。

【0176】ステップS109で、CPUモデル4内で減算命令「(レジスタA) - 04H」を実行し、処理結果をレジスタAに「0H」を格納する。

【0177】ステップS110で、「N」に進み、ステップS101に戻る。

【0178】《ROMアドレス1004Hの処理》3行目と同様に、ステップS101→S102→S103→S201→S202→S301→S302→S307→S104→S105と進む。

【0179】ステップS106で、解説結果「BZ ! 1006H」は、周辺回路読み書きに関係ないので、「N」へ移る。

【0180】ステップS109で、CPUモデル4は、分岐命令「BZ !1006H」を実行する。ここでは、条件（レジスタA-04H=0）が成立するので、1006H番地へジャンプする。すなわち、CPUモデル4は、プログラムカウンタPCに1006Hをセットする。もし、キー入力がないなど条件が成立しないときは、1005H番地の処理が実行されて、1002H番地へジャンプする。

【0181】《ROMアドレス1006Hの処理》ステップS101で、CPUモデル4は、プログラムカウンタの値「1006H」を取得し、ステップS102→S103と進む。

【0182】ステップS104で、前述と同様な動作で、CPUモデル4は、戻り値：5AB5（MOV A, READTIME）を取り込む。

【0183】ステップS105で、CPUモデル4は、戻り値「5AB5H」を解析する。

【0184】ステップS106で、周辺回路に関係のある命令なので、「Y」へ進む。

【0185】ステップS107で、CPUモデル4は、機能関数IORD（READTIME）を生成し、機能関数I/Fモデル5に渡す。なお、READTIMEは、I/Oアドレスを示すラベルである。

【0186】ステップS108で、図10のステップS201に移る。ステップS201で、機能関数I/Fモデル5は、機能関数IORD（READTIME）が機能関数選択テーブル3（図6）の判定条件を満たすので、「Y」に進む。

【0187】ステップS202で、仮想モデルシミュレーションを実行する。図11に移る。

【0188】図11のステップS301で、仮想モデル7は、仮想関数変換テーブル（図7）により、IORD（READTIME）→R_TMに変換し、仮想時計モデルV72（図4）に渡す。

【0189】ステップS302で、仮想関数R_TMはMEM関数でないため「N」へ進む。

【0190】ステップS303で、「Y」へ進む。

【0191】ステップS308で、仮想モデル7は、仮想時計モデルV72（図4）より時刻情報を読み出し、機能関数I/Fモデル5に時刻情報「#10H」を渡し、機能関数I/Fモデル5は、CPUモデル4に時刻情報「#10H」を返す。図10のステップS202に戻り、さらに、図9のステップS108に戻る。

【0192】図9のステップS109で、CPUモデル4は、時刻情報「#10H」をレジスタAに格納する。

【0193】ステップS110で、「N」に進み、ステップS101に戻る。

【0194】《ROMアドレス1007Hの処理》3行目および4行目と同様に、CPUモデル4内部で処理は完了する命令であるから、ステップS101→S102

→S103→S201→S202→S301→S302→S307→S104→S105→S106→S109と進み、レジスタAに「0」を格納する。その後、ステップS110→S101に戻る。

05 【0195】《ROMアドレス1008Hの処理》3行目、4行目および7行目と同様に、CPUモデル4内部で処理は完了する命令であるから、ステップS101→S102→S103→S201→S202→S301→S302→S307→S104→S105→S106→S109と進み、ステップS109で、分岐命令を実行し、「100AH」へジャンプする。

10 【0196】《ROMアドレス100AHの処理》ステップS101で、PCの値「100AH」を取得し、ステップS102→S103→S104→S105→S106→S107と進み、CPUモデル4は、機能関数IOWR（PORTC, #01H）を生成する。

15 【0197】次に、ステップS108と進み、図10に移り、ステップS201→S202と進み、図11のステップS301で、仮想モデル7は、機能関数IOWR（PORTC, #01H）を仮想関数LMP01に変換する。

20 【0198】ステップS302→S308と移り、ステップS308で、仮想モデル7は、仮想関数変換テーブル（図8）をもとに仮想関数LMP01を「1」に変換して、仮想LED表示モデルV74（図4）に渡す。仮想LED表示モデルV74は、LED表示GUI182に渡し、表示装置（図2）にランプの点灯画像を表示させる。

25 【0199】図11の処理を終了し、図10ステップS202に戻り、さらに図9のステップS108に戻る。

30 【0200】ステップS109→S110と進み、ステップS101に戻る。

35 【0201】《ROMアドレス100BHの処理》ステップS101で、PCの値「100BH」を取得し、ステップS102→S103→S201→S202→S301→S302→S307と進み、戻り値「5A03H」（CALL KEYIN）を得て、ステップS103に戻る。

40 【0202】ステップS104で、CPUモデル4は、戻り値「5A03H」（CALLKEYIN）を読み込み、ステップS105→S106→S109と進む。

45 【0203】ステップS109で、CPUモデル4は、レジスタA他の内容をスタックに待避し、プログラムカウンタPCにサブルーチンの先頭アドレス3001H（KEYIN）を設定して、ステップS110を通してS101に戻る。

50 【0204】《ROMアドレス3001Hの処理》ステップS101で、PCの値「3001H」を取得し、ステップS102→S103→S104と進む。

【0205】ステップS104（命令読込）で、通常の

キースキャン・サブルーチン処理であれば、命令コード「A201」（図5）を戻り値として読み込む。しかし、ここでは、図8のアドレス3001H～301FHに示すようなキースキャン・サブルーチン処理を仮想モデルで置き換えて処理するため、予め、アドレス3001Hの内容をダミーの入出力命令「CALL（KEYIN）」に置き換えてあるものとする。あるいは、ラベルKEYINの示すアドレスを他のアドレスに置き換えて、そのアドレスにダミーの入出力命令「CALL（KEYIN）」を書いておいてもよい。

【0206】ステップS105→S106→S107と進む。

【0207】ステップS107（機能関数生成）で、CPUモデル4は、機能関数CALL（KEYIN）を生成し、これを機能関数I/Fモデル5に渡す。

【0208】ステップS108で、周辺回路処理サブルーチン呼出しステップS201で、機能関数I/Fモデル5は、機能関数選択テーブル3と比較。登録されており、かつ時刻が「10H」であり、引数の条件を満たすので、「Y」に進む。

【0209】ステップS202で、仮想モデルシミュレーションを実行する。

【0210】ステップS301で、機能関数I/Fモデル5は、仮想関数変換テーブル（図7）により、機能関数CALL（KEYIN）を仮想関数「K_IN」に変換する。

【0211】ステップS302で、「N」へ
ステップS303で、「N」へ
ステップS304で、「Y」へ
ステップS309で、仮想モデル7は、仮想キー入力モデルV73を起動し、キー入力処理を実行。キー入力GUI181により、あるキーが押下されると、キーの文字情報を取得。仮想モデル7は、キーの文字情報「#ECH」を機能関数I/Fモデル5に渡す。

【0212】ステップS109で、機能関数I/Fモデル5は、CPUモデル4に戻り値「#ECH」を返す。CPUモデル4は、戻り値「#ECH」をレジスタAに格納する。

【0213】《ROMアドレス100CHの処理》3行目、4行目、7行目および8行目と同様に、ステップS109で、CPUモデル4は、レジスタAの値から「ECH」を減算し、演算結果「0H」をレジスタAに格納する処理を実行する。CPUモデル4内で処理は完了し、ステップS101に戻る。

【0214】《ROMアドレス100DHの処理》3行目、4行目、7行目、8行目、12行目および13行目と同様に、マイコンシミュレータ1内部で処理は完了し、「100FH」へジャンプする。

【0215】《ROMアドレス100FHの処理》ステップS101で、PCの値「100FH」を取得し、S

102→S103→S201→S202と進み、戻り値「C700H」（MOV PORTC, #00H）を取得する。

【0216】ステップS105→S106→S107と進み、機能関数IOWR（PORTC, #00H）生成する。

【0217】ステップS108→S201→S202と進み、ステップS301で、仮想モデル7は、機能関数IOWR（PORTC, #00H）を仮想関数LMP00に変換する。

【0218】ステップS302→S303→S304→S305→S310と進む。

【0219】ステップS310で、仮想モデル7内の仮想LED表示モデルV74は、仮想関数LMP00をもとに、LED表示GUI182に「0」を出力して図2の表示装置に消灯画像を表示させる。

【0220】以上で、図3にその具体例を示したハードウェアモデル20についての第1実施例の説明を終えるが、図1に示した第1実施例は、図3の電子手帳にとどまらず、マイコンを使用するなら、その他の電子機器についても適用できるのは言うまでもない。

【0221】本実施例の構成を採用することにより、以下の効果が得られる。

【0222】第1に、シミュレーションに要する時間を短縮できる。これは、プログラムメモリモデルやデータメモリモデルだけでなく、メモリ以外の周辺回路まで仮想モデルを使用して入出力処理を実行するため、ハードウェアシミュレータ2を使用して周辺回路との読み書きする場合に比べ、シミュレーション結果を取得するまでの時間を大幅に短縮できる。例えば、ハードウェアモデルシミュレーションで1マシンサイクルに10msかかるのとすると、4マシンサイクルには40msかかる。これに対して、仮想モデルでは、数μsで同等の結果が得られるので、ロード/ストア命令をシミュレーションする時間を約10,000分の1に低減できる。

【0223】第2に、仮想モデル7を使用できる周辺回路を予め機能関数選択テーブル3に登録しておくことにより、仮想モデルシミュレータ7かハードウェアシミュレータ2のどちらのモデルを使うかが即座に選択できる。従って、シミュレーションの初期段階で、ハードウェアモデルの接続や機能が正しく動作することが確認できれば、機能関数選択テーブル3の内容を書き換えるだけで、仮想モデルを使用して検証ができるようになる。選択できるアドレスや時間などいろいろな条件を設定できるので、用途に応じて仮想モデルを使い分けしやすくなる。

【0224】第3に、機能関数選択テーブル3には他に影響を及ぼさない周辺回路モデルのみ登録することとしたため、仮想モデル7によるシミュレーション結果が他のハードウェアシミュレーションに影響を及ぼすことが

ない。このため、仮想モデルシミュレータ7を使っても、ハードウェアシミュレータ2を使っても、同様のシミュレーション結果を得ることができる。

【0225】第4に、ターゲットプログラム内のサブルーチン処理を1つの仮想モデルで置き換えて検証できるようにしたので、一層処理時間を短縮することができる。本実施例では、キー入力サブルーチンを仮想キー入力モデルに置き換える例を示したが、これに限定されるものではなく、液晶表示用ドライバ、警告音出力回路、あるいは、通信装置などを制御するサブルーチンも仮想モデルに置き換えることができる。例えば、液晶表示用ドライバを使って文字を表示するためには、文字コードを文字パターンに変換し、X方向のドライバとY方向のドライバにそれぞれ文字パターンデータを所定のタイミングで供給し、液晶表示装置上に表示させることになる。これをハードウェアシミュレータ2を使って検証しようとする、非常に複雑で、長いステップのターゲットプログラムを処理することになり、1文字をキー入力モデルから入力して、この文字データを液晶表示装置に表示させるだけで数分を要してしまう。この機能を液晶表示装置と等価な表示ができる仮想モデルで代替えることで、実使用状況と同程度の速度で表示装置に表示させることができる。

【0226】〔第2実施例〕第1実施例は、ハードウェアシミュレーションによって得られる結果と等価な結果になるような機能を予め仮想モデルに登録しておき、ロード/ストア命令を実行するときに、仮想モデルを使用することにより、システムシミュレーションの高速化を図ったものであった。したがって、仮想モデルに登録するモデルの種類が多いほど望ましい。しかしながら、1つの周辺回路のシミュレーション結果が他の周辺回路のシミュレーションに影響を及ぼす場合には、ハードウェアシミュレータで検証した後で、仮想モデルで検証したり、逆に、仮想モデルで検証した後で、ハードウェアシミュレーションを実行すると、その結果が矛盾することになることがある。

【0227】例えば、出力ポートと入力ポートとの間にスイッチが接続されているとする。出力ポートの値を「1」として、入力ポートの値を読み取ることで、スイッチがONしているか、OFFしているかが分かる。即ち、入力ポートの値が「1」であればスイッチはONしており、「0」であればOFFしていることが分かる。

【0228】いま、仮想モデルを使用して出力ポートの値を「1」として、ハードウェアモデルを使って入力ポートの値を読み取るとする。第1実施例あるいは従来のシステムシミュレータでは、仮想モデルとハードウェアモデルとはなんら関連性がないので、仮想モデルの出力ポートの値を「1」にしたからといって、ハードウェアモデルの出力ポートが「1」になっているとは限らない。仮に、ハードウェアモデルの出力ポートが「0」に

なっているとすると、たとえスイッチがONしていたとしても、入力ポートの値は「0」であり、スイッチがOFFしていると誤判定されてしまうことになる。

【0229】そこで、第1実施例では、上述のような不具合が生じないようにするため、シミュレータ結果が互いに関連するモデルは、機能関数選択テーブル(図6)で選択できないようにしたり、それらをまとめて選択するように登録を工夫している。

【0230】第2実施例は、あるモデルのシミュレーション結果が他のモデルのシミュレーションに影響を及ぼすような場合であっても、正しいシミュレーション結果が得られるような構成を採ることにより、システムシミュレーションをより高速化したものである。

【0231】図13は、本発明のシステムシミュレータの第2実施例の機能ブロック図である。図13を図1と対比すれば、第2実施例は第1実施例にデータ更新I/Fモデル9を付加した点が異なっているのがわかる。データ更新I/Fモデル9は、仮想モデル71のシミュレーション結果をハードウェアモデル21に反映し、またハードウェアモデル21のシミュレーション結果を仮想モデル71に反映させる役割を担う。このような更新機能を設けることにより、1つの周辺回路のシミュレーション結果が他の周辺回路のシミュレーションに影響を及ぼす場合であっても、仮想モデルシミュレーションとハードウェアシミュレーションとの間の矛盾を回避することができるのである。

【0232】第2実施例についても、図2に示したシミュレーション装置のブロック図およびその説明は第1実施例におけるのと同様に適用できる。したがって、第2実施例のシステムシミュレータによるシステムシミュレーションを実行するときには、上記シミュレーション装置の準備と、マイコンシミュレータ1-1およびハードウェアシミュレータ2-1のロードが前提となる。

【0233】図14は、第2実施例におけるハードウェアモデル21の構成図であり、HPMテーブルH302付きのプログラムメモリモデルH301と、HDMテーブルH304付きのデータメモリモデルH303と、HIP入力レジスタH310付きの入力ポートモデルH309と、HKD出力レジスタH313付きの出力ポートモデルH311およびデコーダモデルH312とから成る。

【0234】プログラムメモリモデルH301、データメモリモデルH303、入力ポートモデルH309、出力ポートモデルH311は、システムI/Fモデル8と接続され、HPMテーブルH302、HDMテーブルH304、HIP入力レジスタH310、HKD出力レジスタH313は、データ更新I/Fモデル9に接続され、それぞれプログラムメモリモデルH301、データメモリモデルH303、入力ポートモデルH309、出力ポートモデルH311およびデコーダモデルH312

の更新のために使用されるとともに、仮想モデル71の更新のために出力される。なお、HPMテーブルH302は、プログラムメモリの内容に相当し、ROMのようにその内容が更新されることがないわかっている場合には、データ更新I/Fモデル9と接続しなくてもよい。また、入力ポートモデルH309、出力ポートモデルH311およびデコーダモデルH312は、図3におけるI/OポートAモデル12、I/OポートBモデル14およびデコーダモデル15に相当する。

【0235】HPMテーブルH302は、ターゲットプログラムの命令コードを有しており、図5に示すようなシンボルと戻り値とが対になったデータを記憶手段に記憶している。プログラムメモリモデルH301は、機能関数I/Fモデル5が出力する機能関数に基づき、所定のアドレスの命令コードをHPMテーブルH302から読み出して、機能関数I/Fモデル5を介してCPUモデル4に命令コードを返す。

【0236】同様に、HDMテーブルH304は、データメモリのアクセスアドレスと記憶データとが対になったテーブル（記憶手段）を有しており、データメモリモデルH303は、機能関数に基づき、所定のアドレスの記憶データをHDMテーブルH304に書き込んだり、読み出して、機能関数I/Fモデル5を介してCPUモデル4との間でデータの授受を行う。データメモリモデルH303は、テーブルの記憶内容が書き換えられると、データ更新I/Fモデル9を介して仮想モデル71内のVDMテーブルV404の内容を同一にすべく更新する。

【0237】出力ポートモデルH311は、CPUモデル4から機能関数I/Fモデル5を介して供給されたポート出力信号をデコーダモデルH312に出力し、デコーダモデルH312はポート信号をデコードし、キーマトリクスの列数（または行数）に相当する数のデコード信号を出力する。このデコード信号は、いずれか1つが「1」で他は「0」の信号であって、「1」になる信号の位置を順次変えることで、キースキャンを実行する。HKD出力レジスタH313は、このデコード信号をレジスタ（記憶手段）に保存するとともに、データ更新I/Fモデル9を介して仮想モデル71内のVKD出力レジスタV410の内容を同一にすべく更新する。また、このデコード信号は、キー入力GUI181にも供給される。

【0238】入力ポートモデルH309は、キー入力GUI181のキーマトリクスと接続され、キー入力GUI181から入力されたキー入力信号を読み込み、機能関数I/Fモデル5を介してCPUモデル4に返す。ポート入力は、キーマトリクスの行数（または列数）に相当する数だけのポートを有し、ポート入力処理は、列数（または行数）回の入力処理を繰り返す。HKD出力レジスタH313は、キー入力GUI181から入力され

たキー入力信号を一旦レジスタ（記憶手段）に保存するとともに、データ更新I/Fモデル9を介して仮想モデル71内のVIP入力レジスタV408の内容を同一にすべく更新する。

05 【0239】図15は、第2実施例における仮想モデル71の構成図であり、VPMテーブルV402付きの仮想プログラムメモリモデルV401、VDMテーブルV404付きの仮想データメモリモデルV403、VIP入力レジスタV408付きの仮想入力ポートモデルV407およびVKD出力レジスタV410付きの仮想キーデコーダモデルV409から成る。

【0240】仮想プログラムメモリモデルV401、仮想データメモリモデルV403、仮想入力ポートモデルV407、仮想キーデコーダモデルV409は、機能関数I/Fモデル5と接続され、VPMテーブルV402、VDMテーブルV404、VIP入力レジスタV408、VKD出力レジスタV410は、データ更新I/Fモデル9に接続され、それぞれ仮想プログラムメモリモデルV401、仮想データメモリモデルV403、仮想入力ポートモデルV407、仮想キーデコーダモデルV409の更新のために使用されるとともに、ハードウェアモデル21の更新のために出力される。

【0241】図4における仮想キー入力モデルV73は、キー入力GUI181から、文字情報を得る機能を構成していたが、図15における仮想入力ポートモデルV407および仮想キーデコーダモデルV409は、キーマトリクス部分の入力信号と出力信号のレベル（「ハイレベル」または「ロウレベル」）を取り扱う点において、第1実施例の仮想キー入力モデルV73とは大きく異なる。

【0242】VPMテーブルV402は、ターゲットプログラムの命令コードを有しており、図5に示すようなシンボルと戻り値とが対になったデータをテーブル（記憶手段）に記憶している。仮想プログラムメモリモデルV401は、機能関数I/Fモデル5が出力する機能関数に基づき、所定のアドレスの命令コードをVPMテーブルV402から読み出して、機能関数I/Fモデル5を介してCPUモデル4に命令コードを返す。

【0243】同様に、VDMテーブルV404は、データメモリのアクセスアドレスと記憶データとが対になったテーブル（記憶手段）を有しており、仮想データメモリモデルV403は、機能関数に基づき、所定のアドレスの記憶データをVDMテーブルV404に書き込んだり、読み出して、機能関数I/Fモデル5を介してCPUモデル4との間でデータの授受を行う。仮想データメモリモデルV403は、テーブルV404の記憶内容が書き換えられると、データ更新I/Fモデル9を介してハードウェアモデル21内のHDMテーブルH304の内容を同一にすべく更新する。

50 【0244】仮想キーデコーダモデルV409は、CP

Uモデル4から機能関数I/Fモデル5を介して供給されたポート出力信号をデコーダをデコードし、キーマトリクスの列数（または行数）に相当する数のデコード信号を出力する。このデコード信号は、いずれか1つが「1」で他は「0」の信号であって、「1」になる信号の位置を順次変えることで、キースキャンを実行する。VKD出力レジスタV410は、このデコード信号をレジスタ（記憶手段）に保存するとともに、データ更新I/Fモデル9を介してハードウェアモデル21内のHKD出力レジスタH313の内容を同一にすべく更新する。また、このデコード信号は、キー入力GUI181にも供給される。

【0245】ハードウェアモデル21は、出力ポートモデルH311とデコーダモデルH312との接続情報とこれに輸入される端子信号をもとに演算を行い、各内部回路に信号を伝搬させ、出力端子まで到達させることでデコード信号を得る。このため、出力ポートモデルH311とデコーダモデルH312とは、デコード信号を得るのに数10～数100msの時間を要する。これに対して、仮想モデル71は、出力ポートに出力するデータをもとに論理演算するか、変換テーブルによりデコード信号を得るようにしている。仮想モデル71内の仮想キーデコーダモデルV409は、例えば、ポート出力データ（0, 1）をデコード信号（0, 0, 1, 0）に変換するだけであるので、数μs程度の極めて短時間で結果を得ることができる。

【0246】仮想入力ポートモデルV407は、キー入力GUI181のキーマトリクスと接続され、キー入力GUI181から入力されたキー入力信号を読み込み、機能関数I/Fモデル5を介してCPUモデル4に返す。ポート入力は、キーマトリクスの行数（または列数）に相当する数だけのポートを有し、ポート入力処理は、列数（または行数）回の入力処理を繰り返す。VIP入力レジスタV408は、キー入力GUI181から入力されたキー入力信号を一旦レジスタ（記憶手段）に保存するとともに、データ更新I/Fモデル9を介してハードウェアモデル21内のHIP入力レジスタH310の内容を同一にすべく更新する。

【0247】仮想入力ポートモデルV407は、キー入力信号をそのままCPUモデル4に返すだけなので数μsで処理が終わる。これに対して、ハードウェアモデル21内の入力ポートモデルH309は、接続情報とこれに輸入される端子信号をもとに演算を行い、各内部回路に信号を伝搬させ、出力端子まで到達させることでキー入力信号をCPUモデル4に返すので、処理が終わるのに数10msもかかる。

【0248】図16は、機能関数選択テーブル31の内容例であり、機能関数I/Fモデル5が、仮想モデル71を選択可能な機能関数とその判断条件を示す。ここで、PORTBは、出力ポートモデルH311のI/O

アドレスを表すラベルである。これ以外の本テーブル中の引数とその意味は図6についての説明がそのまま当てはまるため説明は省略する。機能関数選択テーブル31の内容は、ハードウェアシミュレーションの対象を何にするか、何を仮想モデルで置き換えられるかによって決まり、常に固定されているものではない。また、機能関数選択テーブル31には、仮想モデルに登録されていない機能関数を登録することはできないが、仮想モデルに登録されているモデルに対応する機能関数を全て登録する必要はない。

【0249】図17は、第2実施例における周辺回路処理サブルーチンの詳細フローチャートを示し、図9における周辺回路処理サブルーチンS103, S108によって呼び出されるものである。同図中、実線は処理の流れ、点線はデータの流れを示す。

【0250】第1実施例と同様、CPUモデル4が、命令コードをフェッチしたり、ロード命令やストア命令など周辺装置との入出力を伴う命令を実行した場合、周辺回路処理サブルーチンが呼び出される。以下、図13を参照して、図17の流れを説明する。

【0251】図17において、CPUモデル4が機能関数I/Fモデル5に機能関数を渡すと、ステップS401で、機能関数I/Fモデル5は、機能関数選択テーブル31を参照して、仮想モデルシミュレーション（S402）を使うか、ハードウェアシミュレーション（S409）を使うか判断する。機能関数I/Fモデル5は、CPUモデル4から受け取った機能関数が機能関数選択テーブル31に登録されていて、かつその判断条件を満たしていれば、「Y」へ進み、仮想モデルシミュレーション（S402）を実行し、機能関数が登録されていないか、または判断条件を満たしていなければ、「N」へ進み、バスI/Fモデルシミュレーション（S403）に移る。

【0252】ステップS402（仮想モデルシミュレーション）で、仮想モデル71は、後述の仮想モデルシミュレーションを実行する。実行終了後、シミュレーション結果を各モデルに付随するファイルやレジスタなどの記憶手段に保存するとともに、機能関数I/Fモデル5を介してCPUモデル4にその結果を返す。

【0253】ステップS420（ハードウェアデータ更新起動）で、仮想モデル71は、仮想モデルシミュレーションを実行後、必要に応じてハードウェアデータ更新起動を行い、ハードウェアモデル21に必要なデータを供給する。この結果、ハードウェアモデル21は、所定のファイルやレジスタのデータ更新を行い（S423）、仮想モデルシミュレーションの結果をハードウェアモデル21に反映する。ここで、プログラムメモリモデルH301のように、データ更新する必要がないものについては、初めから本ステップをスキップするようにしてもよい。

【0254】一方、ステップS401において、「N」に進み、ハードウェアシミュレータ2-1を使う場合には、ステップ403に移る。ステップS403（バスI/Fモデルシミュレーション）で、バスI/Fモデル6は、機能関数I/Fモデル5から機能関数を受領し、ハードウェアモデルシミュレーションに必要な端子信号に変換する。なお、バスI/FモデルシミュレーションS403の詳細は図10におけるバスI/FモデルシミュレーションS203に示すとおりである。また、端子信号については、第1実施例の図12と同じである。

【0255】ステップS410（シミュレータ間I/F処理）で、バスI/Fモデル6は、前記端子信号名や数値、コードなどをハードウェアシミュレータ2-1で使用する信号端子名や、数値、コードなどに交換したり、信号授受のタイミングを調整する。なお、マイコンシミュレータ1-1とハードウェアシミュレータ2-1で信号端子名などのコード体系が同一である場合には、本ステップをスキップするようにしてもよい。

【0256】ステップS411（システムI/Fモデルシミュレーション）で、システムI/Fモデル8は、バスI/Fモデル6より、前記端子信号を受領し、ハードウェアモデル21に渡す。なお、システムI/FモデルシミュレーションS411の詳細は図10におけるシステムI/FモデルシミュレーションS211に示すとおりである。

【0257】ステップS409（ハードウェアシミュレーション）で、ハードウェアモデル21は、入力された端子信号とハードウェアモデル21の接続情報に基づき、モデル内の接続点の状態を演算によって求め、出力点の状態がどのようになるかを求める。ハードウェアモデル21は、このようにして求めた結果を、システムI/Fモデル8に返す。

【0258】ステップS424（仮想モデル書き替え必要？）で、ハードウェアモデル21は、機能関数選択テーブル31を参照して、仮想モデル71に付随する所定のファイルやレジスタのデータ更新を行う必要があるかを判断する。S409で処理したハードウェアシミュレーションが、機能関数選択テーブル31に登録されていて、かつ引数が判断条件を満たす場合には、ハードウェアモデル21は、「Y」へ進み、ステップS425の処理に移る。逆に、S409で処理したハードウェアシミュレーションが、機能関数選択テーブル31に登録されていないか、または引数が判断条件を満たさない場合には、ハードウェアモデル21は、「N」へ進み、ステップS411の処理に移る。

【0259】仮想モデル71に登録されているサブモデルは、ハードウェアモデル21に必ず存在するので、データ更新起動（S420）を処理するとき、書き替えの必要性を判断する必要はなかった。しかし、ハードウェアモデル21に登録されているサブモデルは、仮想モデ

ル71に必ずしも存在するとは限らないので、データ更新起動（S425）を処理する前に、書き替えの必要性を判断する処理（S424）が必要となる。

【0260】ステップS425（仮想モデルデータ更新起動）で、ハードウェアモデル21は、ハードウェアモデルシミュレーション（S409）の結果を仮想モデル71に渡し、仮想モデル71は、所定のファイルやレジスタのデータ更新を行い、前記結果を反映する。

【0261】ステップS411（システムI/Fモデルシミュレーション）で、システムI/Fモデル8は、ハードウェアシミュレーション（S409）の結果をステップS410の処理を介してバスI/Fモデル6に返す。

【0262】ステップS403（バスI/Fモデルシミュレーション）で、バスI/Fモデル6は、ハードウェアシミュレーション（S409）の結果を機能関数I/Fモデル5を介してCPUモデル4に渡す。

【0263】以上の処理が終了すると、本サブルーチンと呼び出したステップS103またはS108（図9）に戻る。

【0264】図18は、第2実施例における仮想モデルシミュレーションS402（図17）の詳細フローチャートを示す。

【0265】図18において、ステップS601で、仮想モデル71は、機能関数I/Fモデル5から機能関数を受け取ると、図7と同様、機能関数を仮想モデル71のコード体系にあうように仮想関数に変換する。例えば、命令フェッチ関数FETCH(m)はMEMmに、データメモリ読み書き関数MEMRW(m)はDMEに、ポートA入力関数IORD(PORTA)はIPTaに、ポートB出力関数IOWR(PORTB, n)はKDCnにそれぞれ変換される。なお、コード体系が同じであれば、本ステップは省略してもよい。

【0266】ステップS602で、仮想モデル71は、仮想関数がFETCH関数MEMmであるかを判断し、MEMmであれば、「Y」へ進み、ステップS603へ移る。MEMmでなければ、「N」へ進み、ステップS604へ移る。

【0267】ステップS604で、仮想モデル71は、仮想関数がデータメモリ読み書き関数DMEmであるかを判断し、DMEmであれば、「Y」へ進み、ステップS605へ移る。DMEmでなければ、「N」へ進み、ステップS610へ移る。

【0268】ステップS610で、仮想モデル71は、仮想関数がポートA入力関数IPTaであるかを判断し、IPTaであれば、「Y」へ進み、ステップS611へ移る。IPTaでなければ、「N」へ進み、ステップS612へ移る。

【0269】ステップS612で、仮想モデル71は、仮想関数がポートB出力関数KDCnであるかを判断

し、KDCnであれば、「Y」へ進み、ステップS613へ移り、KDCnでなければ、「N」へ進み、ステップS614へ移る。

【0270】ステップS614で、仮想モデル71は、受領した仮想関数が登録されていない関数であったり、パラメータが違っていたりすると、表示装置にエラーメッセージを表示する。

【0271】ステップS603で、仮想モデル71内の仮想プログラムメモリモデルV401(図15)は、テーブルを参照して、m番地の命令コードをCPUモデル4に返す。テーブルは、図5に示すターゲットプログラムのテーブルに相当し、これを予めVPMテーブルV402に登録してある。仮想プログラムメモリモデルV401は、仮想関数を示すシンボルMEMmを受け取ると、このシンボルに対応する戻り値を読み出し、機能関数I/Fモデル5を介してCPUモデル4に返し、処理を終了する。

【0272】通常、VPMテーブルV402のプログラム内容は書き替わることがないので、プログラムメモリモデルH301のHPMテーブルH302の内容を更新する必要がない。従って、図17では、ステップS603終了後、ステップS420を介さずに仮想モデルシミュレーションS402の処理を終了するようにしている。

【0273】ステップS605で、仮想モデル71内の仮想データメモリモデルV403(図15)は、CPUモデル4とデータを授受し、テーブル内の指定の場所にデータを読み書きする。テーブルは、データメモリモデルの記憶容量に相当する行の記憶領域を有し、テーブルの1つの行はアドレス情報を含むシンボル(DMEM)と読み書きデータとで構成される。VDMテーブルV404には、データメモリモデルの最下位アドレスkから最上位アドレスlまでのシンボルDMEk~DMElが予め付与されている。書き込みの場合、仮想データメモリモデルV403は、CPUモデル4から仮想関数「DEMM,d」を受け取ると、このシンボルに対応する位置DMEmにデータdを書き込む。読み出しの場合には、仮想データメモリモデルV403は、CPUモデル4からデータ情報を含まない仮想関数「DEMM」を受け取ると、このシンボルに対応する位置DMEmからデータdを読み出し、このデータdを機能関数I/Fモデル5を介してCPUモデル4に返す。

【0274】ステップS611で、仮想モデル71内の仮想入力ポートモデルV407(図15)は、キー入力GUI181から押下されたキーマトリクスの出力レベル情報を取得し、一旦VIP入力レジスタV408に保存するとともに、その出力レベル情報を戻り値として機能関数I/Fモデル5を介してCPUモデル4に返し、処理を終了する。一方、VIP入力レジスタV408の内容は、ステップS420の処理に渡され、データ更新

I/Fモデル9(図13)はHIP入力レジスタ(図14)の内容を更新する。

【0275】ステップS613(データ出力・デコード処理)で、仮想モデル71内の仮想キーデコードモデルV409(図15)は、受領した機能関数IOWR(PORTB,D*)をもとに、出力ポートモデルH311とデコードモデルH312(図15)と等価な端子出力レベルが得られるように処理して、この端子出力をキー入力GUI181に供給する。例えば、ポート端子(PB1,PB0)にデータ(d1,d0)を出力する場合、機能関数は、IOWR(PORTB,d1・d0)で表され、仮想キーデコードモデルV409は、次の論理演算を行い、端子出力レベルD3,D2,D1,D0を得る。

【0276】 $D3 = d1 * d0$

$D2 = d1 * /d0$

$D1 = /d1 * d0$

$D0 = /d1 * /d0$

ここで、「/」は信号d1またはd0の論理否定(反転信号)を意味する。

【0277】得られた演算結果は、VKD出力レジスタV410に保存されるとともに、VIP入力レジスタV408の内容は、ステップS420の処理に渡され、データ更新I/Fモデル9(図13)はHKD出力レジスタ(図14)の内容を更新する。

【0278】このように、データ出力・デコード処理S613は、ハードウェアモデルの処理とはことなり、前記演算によってキー入力GUI181のキーマトリクスモデルへ供給する端子出力レベルを求めるので、数 μs の短時間で仮想モデルシミュレーション結果を得ることができる。

【0279】ステップS420(ハードウェアモデルデータ更新起動)で、仮想モデル71は、各テーブルV404、または各レジスタV408,V410の内容を、データ更新I/Fモデル9(図13)に渡す。データ更新I/Fモデル9は、ステップS423(図17)のデータ更新を起動し、ハードウェアモデル21内の仮想モデルに対応する各テーブルH304、または各レジスタH310,H313の内容を更新する。

【0280】以上の仮想モデルシミュレーションS402の処理が終了すると、マイコンシミュレータ1-1は、図17の処理に戻り、ステップS402を終了して、更に図9のサブルーチン呼び出し元S103またはS108へ戻る。

【0281】図19は、図17のうち1点鎖線で示された部分、即ち、ハードウェアシミュレーションS409と仮想モデルデータ更新S424,S425の詳細フローチャートを示す。

【0282】まず、ハードウェアモデル21は、システムI/Fモデル8から端子信号を受け取ると、S801

の処理に移る。

【0283】ステップS801 (MEM=0?) で、ハードウェアモデル21 (図13) は、端子信号MEMを確認して、プログラムメモリモデルH301またはデータメモリモデルH303の呼び出し可否かを判断する。端子信号MEMが「0」であれば、ハードウェアモデル21は、「Y」へ進み、ステップS802の処理に移り、「1」であれば、「N」へ進み、ステップS804の処理に移る。

【0284】ステップS802 (I/O=0?) で、ハードウェアモデル21 (図13) は、端子信号I/Oを確認して、メモリを除く周辺回路H309、H311 (図14) の呼び出し可否かを判断する。端子信号I/Oが「0」であれば、ハードウェアモデル21は、「Y」へ進み、ステップS807の処理に移り、「1」であれば、「N」へ進み、ステップS809の処理に移る。

【0285】メモリ関連の処理である場合、ステップS802の処理に移る。

【0286】ステップS802 ($0 \leq \text{ADRS} < 2000$?) で、ハードウェアモデル21 (図13) は、端子信号ADRSを確認して、アドレス信号が0000H番地以上で2000H番地未満かを判断する。端子信号ADRSが前記範囲内であれば、ハードウェアモデル21は、プログラムメモリモデルH301 (図14) の読込処理であると判断して「Y」へ進み、ステップS810の処理に移り、範囲外であれば、「N」へ進み、ステップS803の処理に移る。

【0287】ステップS803 ($\text{ADRS} \geq 2000$?) で、ハードウェアモデル21 (図13) は、端子信号ADRSを確認して、アドレス信号が2000H番地以上かを判断する。端子信号ADRSが前記範囲内であれば、ハードウェアモデル21は、データメモリモデルH303 (図14) の読み書き処理であると判断して「Y」へ進み、ステップS811の処理に移り、範囲外であれば、「N」へ進み、ステップS809の処理に移る。

【0288】メモリ以外の周辺回路関連の処理である場合、ステップS807の処理に移る。

【0289】ステップS807 (IPT?) で、ハードウェアモデル21 (図13) は、端子信号ADRSを確認して、アドレス信号がPORTA番地かを判断する。端子信号ADRSがPORTA番地であれば、ハードウェアモデル21は、入力ポートモデルH309 (図14) の読み込み処理であると判断して「Y」へ進み、ステップS814の処理に移り、PORTA以外であれば、「N」へ進み、ステップS808の処理に移る。

【0290】ステップS808 (OPT?) で、ハードウェアモデル21 (図13) は、端子信号ADRSを確認して、アドレス信号がPORTB番地かを判断す

る。端子信号ADRSがPORTB番地であれば、ハードウェアモデル21は、出力ポートモデルH309 (図14) の書き込み処理であると判断して「Y」へ進み、ステップS815の処理に移り、PORTB以外であれば、「N」へ進み、ステップS809の処理に移る。

【0291】ステップS801~S804、S807、S808で何れも「N」であった場合、ステップS809 (エラー処理) で、ハードウェアモデル21 (図13) は、該当処理が存在しない旨を表示装置 (図2) に表示するとともに、システムI/Fモデル8にエラー信号「ERR」を渡して、処理を終了する。

【0292】一方、ステップS810 (プログラム読込処理) で、プログラムメモリモデルH301は、HPMテーブルH302 (図14) を参照して、アドレス端子信号ADRSに対応する命令コードを読み出し、命令コードをシステムI/Fモデル8に返す。ここで、プログラムメモリモデルH301は、接続情報に基づき入力された端子信号を各接続点に伝搬させ、出力端子の信号レベル (命令コード) を演算処理して求める。このため、回路規模が大きいほど、演算結果が出るのに時間がかかる。

【0293】プログラムメモリモデルH301内のHPMテーブルH302は、シミュレーションの最中に内容が書き変わることがないので、仮想モデル71のデータ更新をする必要はなく、S810の処理が終了すると、図17のステップS409に戻り、S411の処理に移る。

【0294】ステップS811 (データメモリ読み書き処理) で、データメモリモデルH303は、システムI/Fモデル8との間で授受する記憶データをHDMテーブルH304 (図14) に読み書きする。書き込みの場合、データメモリモデルH303は、アドレス端子信号ADRS番地とこれに対応する記憶データDATAをシステムI/Fモデル8から受け取り、HDMテーブルH304内の所定の領域に記憶する。読み出しの場合、データメモリモデルH303は、アドレス端子信号ADRS番地をシステムI/Fモデル8から受け取り、HDMテーブルH304内の所定の領域から記憶データDATAを読み出して、システムI/Fモデル8に返す。ここで、データメモリモデルH303は、接続情報に基づき入力された端子信号を各接続点に伝搬させ、出力端子の信号レベル (記憶データ) を演算処理して求める。本処理終了後は、ステップS424に移る。

【0295】ステップS815 (データ出力・デコード処理) で、出力ポートモデルH311 (図14) は、システムI/Fモデル8から、アドレス信号PORTB、データDATAなどを受け取り、接続情報に基づいて、ポート出力信号を求める。デコードモデルH312 (図14) は、ポート出力信号を入力信号として、接続情報に基づいて、デコード出力信号を求める。このデコード

出力信号は、HKD出力レジスタH313に保存されるときともに、キー入力GUI181と、データ更新I/Fモデル9に供給される。この結果、仮想モデル71内のVKDレジスタV410(図15)の内容もHKD出力レジスタH313の内容と等価になる。本処理終了後は、ステップS424に移る。

【0296】ステップS814(ポート入力処理)で、入力ポートモデルH309(図14)は、キー入力GUI181からキーマトリクスの出力に相当する信号を受け取り、HIPレジスタH310(図14)に保存するときともに、データ更新I/Fモデル9に供給する。この結果、仮想モデル71内のVIPレジスタV408(図15)の内容もHIP入力レジスタH310の内容と等価になる。

【0297】また、HIPレジスタH310の内容は、CPUモデル4の読み出し命令によって、システムI/Fモデル8を介してCPUモデル4に返される。この処理も、上述と同様、接続情報に基づいて行われる。本処理終了後は、ステップS424に移る。

【0298】ステップS424(仮想モデル書替?)で、データ更新I/Fモデル9は、機能関数選択テーブル31(図13、図16)を参照して、ハードウェアシミュレーションS409の結果を仮想モデル71に反映するか否かを判断する。本実施例では、機能関数IOWR(PORTB, D*, I/O, WR)だけが仮想モデルシミュレーションS402を実行し、データ更新対象であるので、ステップS815の処理を実行した場合だけ「Y」に進み、ステップS425の処理に移る。ステップS811, S814の処理を実行した場合は「N」に進み、ハードウェアモデルシミュレーションを終了する。

【0299】ステップS425(仮想モデルデータ更新起動)で、ハードウェアモデル21は、更新すべきデータをデータ更新I/Fモデル9に供給し、これを仮想モデル71(図13)に渡す。仮想モデル71は、データ更新S421(図17)を起動して、所定のテーブルやレジスタの内容を更新する。

【0300】以上の処理が終了すると、図17のステップS409, S424, S425の処理を終了し、ステップ411の処理に移る。

【0301】図20は、ハードウェアモデルデータ更新S423(図17)の詳細フローチャートを示す。

【0302】仮想モデルシミュレーションS402(図17)を実行した結果、ハードウェアモデルデータ更新起動S420(図18)が処理されると、仮想モデル71は、更新すべきデータと、更新先の情報をデータ更新I/Fモデル9に供給し、図20の開始に移る。

【0303】ステップS501(仮想→ハード・コード変換)で、データ更新I/Fモデル9は、受領した更新データをハードウェアモデルシミュレータ2-1(図1

3)内のコード体系に変換する。例えば、仮想データメモリモデルV403内のVDMテーブルV404(図15)が書き替えられた場合、シンボルDME_mと更新データDATAを「C_HDME、DATA」に変換する。また、HKD出力レジスタH313(図14)が書き替えられた場合、デコード出力信号「D3, D2, D1, D0」を「C_HKDC、D3, D2, D1, D0」に変換する。

【0304】仮想モデルシミュレータ71とハードウェアモデルシミュレータ2-1とでコード体系が同じ場合には、本ステップを省略してもよい。処理終了後、ステップS503に移る。

【0305】ステップS503(C_HDME?)で、データ更新I/Fモデル9(図13)は、受領したコードが「C_HDME」であるか否かを判断する。「C_HDME」であれば「Y」に進み、ステップS509の処理に移る。「C_HDME」でなければ「N」に進み、ステップS507の処理に移る。

【0306】ステップS507(C_HKDC?)で、データ更新I/Fモデル9は、受領したコードが「C_HKDC」であるか否かを判断する。「C_HKDC」であれば「Y」に進み、ステップS513の処理に移る。「C_HKDC」でなければ「N」に進み、ステップS514の処理に移る。

【0307】ステップS514(エラー処理)で、データ更新I/Fモデル9は、受領したコードがいずれにも該当しない場合には、表示装置(図2)にエラー発生の表示を行い、処理を終了する。

【0308】ステップS509で、データ更新I/Fモデル9は、コード「C_HDME、DATA」を受領すると、データメモリモデルH303内のHDMテーブルH304(図14)を書き替え、処理を終了する。

【0309】ステップS513で、データ更新I/Fモデル9は、コード「C_HKDC、D3, D2, D1, D0」を受領すると、出力ポート・デコードモデルH311、H312内のHKDテーブルH313(図14)を書き替え、処理を終了する。

【0310】以上の処理が終了すると、図17のステップS420を終了し、図9のステップS103またはS108に戻る。

【0311】このようにして、仮想モデルシミュレーションS402の結果がハードウェアモデル21のテーブルやレジスタに反映される。なお、仮想モデル71を使用するサブモデルで、データ更新が必要になるものは、図20に示すデータ更新S423の処理に予め登録しておく必要がある。また、本実施例のように、仮想モデルを使わないデータ更新処理、例えばS509を余分に登録しておくこともできる。この結果、ハードウェアモデルデータ更新処理S423の内容を書き替えることなく、機能関数選択テーブル31(図13)の内容を書き

替えるだけで、データメモリモデルを仮想モデルシミュレーションに切り替えた場合にも即応できるようになる。

【0312】図21は、仮想モデルデータ更新S421(図17)の詳細フローチャートを示す。

【0313】ハードウェアモデルシミュレーションS409(図17)を実行した結果、仮想モデルデータ更新起動S425(図19)が処理されると、ハードウェアモデル21(図13)は、更新すべきデータと、更新先の情報をデータ更新I/Fモデル9に供給し、図21の開始に移る。

【0314】ステップS701(ハード→仮想・コード変換)で、データ更新I/Fモデル9は、受領した更新データを仮想モデルシミュレータ71(図13)内のコード体系に変換する。例えば、データメモリモデルH303内のHDMテーブルH304(図14)が書き替えられた場合、アドレス信号ADRSと更新データDATAをシンボル「C_DME」と「DATA」に変換する。また、VKD出力レジスタV410(図15)が書き替えられた場合、デコード出力信号「D3, D2, D1, D0」を「C_KDC, D3, D2, D1, D0」に変換する。

【0315】仮想モデルシミュレータ71とハードウェアモデルシミュレータ2-1とでコード体系が同じ場合には、本ステップを省略してもよい。処理終了後、ステップS703に移る。

【0316】ステップS703(C_DME?)で、データ更新I/Fモデル9(図13)は、受領したコードが「C_DME」であるか否かを判断する。「C_DME」であれば「Y」に進み、ステップS710の処理に移る。「C_DME」でなければ「N」に進み、ステップS707の処理に移る。

【0317】ステップS707(C_KDC?)で、データ更新I/Fモデル9は、受領したコードが「C_KDC」であるか否かを判断する。「C_KDC」であれば「Y」に進み、ステップS714の処理に移る。「C_KDC」でなければ「N」に進み、ステップS708の処理に移る。

【0318】ステップS708(エラー処理)で、データ更新I/Fモデル9は、受領したコードがいずれにも該当しない場合には、表示装置(図2)にエラー発生の表示を行い、処理を終了する。

【0319】ステップS710(VDMテーブル書き替え処理)で、データ更新I/Fモデル9は、コード「C_DME、DATA」を受領すると、仮想データメモリモデルV403内のVDMテーブルV404(図15)を書き替え、処理を終了する。

【0320】ステップS714で、データ更新I/Fモデル9は、コード「C_KDC、D3、D2、D1、D0」を受領すると、仮想キーデコードモデルV409内

のVKDテーブルV410(図15)を書き替え、処理を終了する。

【0321】以上の処理が終了すると、図17のステップS425を終了し、ステップS411に移る。このようにして、ハードウェアモデルシミュレーションS409の結果が仮想モデル71のテーブルやレジスタに反映される。なお、ハードウェアモデル21を使用するサブモデルで、データ更新が必要になるものは、図21に示すデータ更新S421の処理に予め登録しておく必要がある。また、本実施例のように、仮想モデルを使わないデータ更新処理、例えばS710を余分に登録しておくこともできる。この結果、仮想モデルデータ更新処理S421の内容を書き替えることなく、機能関数選択テーブル31(図13)の内容を書き替えるだけで、仮想データメモリモデルをハードウェアモデルシミュレーションに切り替えた場合にも即応できるようになる。

【0322】[ターゲットプログラム例による動作の説明]図5および図8に示したプログラム例により、図17～図21のフローチャートを使用してその動作について説明する。第2実施例の特徴部分のみの説明に止めて第1実施例との重複を避けるため、図8のターゲットプログラム・リストの第1行と第2行について説明することとする。

【0323】第2実施例においても、図9に示したCPUモデル4による処理フローはそのまま適用できる。

【0324】さて、先のプログラム例の第1行は、I/OポートBモデル14がデコーダモデル15に、(PB1, PB0) = (0, 1)を出力し、デコーダモデル15はキーマトリクスモデル13に(D3, D2, D1, D0) = (0, 0, 1, 0)を出力するというものであったが、第2実施例では、これを仮想モデルシミュレーションで行い、その結果をハードウェアモデル21に反映させる。また、先のプログラムの第2行は、キーマトリクスモデル13の状態(PA3, PA2, PA1, PA0) = (0, 1, 0, 0)をI/OポートAモデル12に取り込むというものであったが、これをハードウェアシミュレーションにより行う。以下、各行ごとに詳細に説明する。

【0325】《ROMアドレス1001Hの処理》図9のステップS101で、CPUモデル4は、プログラムカウンタPCの値「1001H」を取得する。

【0326】ステップS102で、CPUモデル4は、機能関数FETCH(1001H)を生成し、機能関数I/Fモデル5に渡す。

【0327】ステップS103で、周辺回路処理サブルーチンが呼び出される。

【0328】図17のステップS401で、機能関数I/Fモデル5(図13)は、機能関数選択テーブル31(図16)に関数が登録されているか、かつその引数の条件を満たしているか、チェックを行う(図17)。FE

TCH関数は登録されていて、アドレスの条件を満たしているので「Y」に進み、機能関数を仮想モデル71に渡す。

【0329】ステップS402で、仮想モデルシミュレーションS402を実行する。詳細フローチャートを図18に示す。

【0330】図18のS601で、仮想モデル71は、機能関数I/Fモデル5から受領した機能関数FETCH(1001H)を、仮想関数変換テーブル(図7相当)により、仮想関数「MEM1001」に変換する。

【0331】ステップS602で、仮想モデル71は、仮想関数「MEM1001」がメモリを読み出すための関数(MEM関数)か判断する。今の場合、MEM関数なので「Y」へ進む。

【0332】S603で、仮想モデル71は、仮想プログラムメモリモデルV401内のVPMテーブルV402(図15)を参照し、ラベル「MEM1001」に対応する戻り値「A201」(図5)を取得し、機能関数I/Fモデル5にこれを返す。なお、この場合は、仮想モデル71に書き込むわけではないため、ハードウェアデータ更新起動(S420)はバイパスされる。

【0333】以上で、仮想モデルシミュレーションS402が終了し、図9に戻る。

【0334】図9のステップS104で、機能関数I/Fモデル5はCPUモデル4に戻り値「A201」を渡す。CPUモデル4は、戻り値「A201」を読み込み、命令レジスタ(不図示)に格納する。

【0335】ステップS105で、CPUモデル4は、戻り値「A201」を解析し、「MOV PORTB, #01H」であると解釈する。

【0336】ステップS106で、CPUモデル4は、解釈した命令が周辺回路の読み書きに関するものかを判断し、「Y」に進む。

【0337】ステップS107で、CPUモデル4は、解釈した命令に対応する機能関数IOWR(PORTB, #01H, I/O, WR)を生成し、機能関数I/Fモデル5に渡す。

【0338】ステップS108で、周辺回路処理サブルーチン呼出す。

【0339】図17のステップS401で、機能関数I/Fモデル5は、仮想モデル71を呼び出すかどうかの判断を行う。IOWR関数は、図16の機能関数選択テーブル31に登録されていて、且つその引数の条件を満たしているので「Y」に進むとともに、機能関数IOWR(PORTB, #01H)を仮想モデル71に渡す。

【0340】ステップS402で、仮想モデルシミュレーションS402を実行する。

【0341】図18のS601で、仮想モデル715は、仮想関数変換テーブル(図7相当)により、機能関数IOWR(PORTB, #01H)を仮想関数KDC

01に変換する。

【0342】ステップS602で、「N」へ進む。

【0343】ステップS604で、「N」へ進む。

【0344】ステップS610で、「N」へ進む。

05 【0345】ステップS612で、「Y」へ進む。

【0346】ステップS613で、仮想モデル71の仮想キーデコーダモデルV409(図15)でポート出力およびデコード処理を論理演算により行い、その結果をVKD出力レジスタV410に保存する。具体的には、第1実施例と同様に、デコード出力(D3, D2, D1, D0) = (0, 0, 1, 0)となる。これをキー入力GUI181に出力し、キーマトリクスモデル13の右から2本目の出力を「1」にする。

15 【0347】ステップS420で、この場合は、仮想モデル71への書き込みを行ったので、ハードウェアモデルデータ更新S423を起動する。

【0348】図20のステップS501で、データ更新I/Fモデル9は、VKD出力レジスタV410から読み出した仮想コードKDC(0010)をハードウェアコード「C_HKDC(0010)」に変換する。

20 【0349】ステップS503で、「N」へ進む。

【0350】ステップS507で、「Y」へ進む。

【0351】ステップS513で、データ更新I/Fモデル9は、ハードウェアモデル21のHKD出力レジスタH313(図14)のデータを更新する。この結果、ハードウェアモデル21のデコーダモデル15の出力(D3, D2, D1, D0) = (0, 0, 1, 0)となり、仮想モデル71側と整合することとなる。

30 【0352】以上で、図20の処理を終了し、更に図18の処理を終了し、図9に戻る。

【0353】図9のステップS109で、CPUモデル4は、機能関数I/Fモデル5からシミュレーション完了信号「DONE」を受領する。

【0354】ステップS110で、CPUモデルシミュレータ4は、「N」へ進み、ステップS101へ戻る。

35 【0355】以上で、プログラムの第1行目のシミュレーションが終了する。

【0356】《ROMアドレス1002Hの処理》図9のステップS101~S106で、PCの値「1002H」取得から、「MOV A, PORTA」との命令解釈までは、第1行目と同様の手順のため説明は省略する。

45 【0357】ステップS107で、CPUモデル4は、解釈した命令に対応する機能関数IORD(PORTA, I/O, RD)を生成し、機能関数I/Fモデル5に渡す。

【0358】ステップS108で、周辺回路処理サブルーチンを呼出す。

50 【0359】図17のステップS401で、この機能関数IORD(PORTA, I/O, RD)は図16の機

能関数選択テーブル31に登録されていないので「N」へ進む。

【0360】ステップS403（バスI/Fモデルシミュレーション）で、バスI/Fモデル6（図13）は、機能関数IORD（PORTA, I/O, RD）を端子信号「ADRS, /CS, /RD」などに変換する。

【0361】ステップS410（シミュレータ間I/F処理）で、バスI/Fモデル6（図13）は、端子信号名をハードウェアシミュレータ2-1対応に変換したり、データ授受のタイミングを調整する。

【0362】ステップS411（システムI/Fモデルシミュレーション）で、システムI/Fモデル8は、端子信号（図12）を受け取る。端子信号の授受動作は第1実施例と同様であるからその説明は省略する。

【0363】ステップS409で、ハードウェアシミュレーションを実行する。

【0364】以下、ハードウェアシミュレーションS409の詳細を説明する。

【0365】図19のステップS801で、ハードウェアモデル21（図13）は、端子信号MEMが「0」か、すなわちメモリの読み書きサイクルかを判断し、「N」へ進む。

【0366】ステップS804で、ハードウェアモデル21は、端子信号I/Oが「0」か、すなわちメモリ以外の周辺回路の読み書きサイクルか判断し、「Y」へ進む。

【0367】ステップS807で、アドレス信号ADRSが「PORTA」か判断し、「Y」へ進む。

【0368】ステップS814（ポート入力処理）で、入力ポートモデルH309は、接続情報に基づき各接続点のレベルを演算によって求める。いま、キー入力GUI181のキーマトリクスの右から2番目で上から3番目のスイッチが押下されているとすると、デコード出力D1は「1」であるので、入力ポートAモデル12の入力PA2も「1」となる。従って、入力ポートAに（PA3, PA2, PA1, PA0）=（0, 1, 0, 0）を得る。入力ポートモデルH309（図14）のデータは、HIP入力レジスタ310に記憶される。

【0369】ステップS424（仮想モデル書替？）で、ハードウェアシミュレーションS409の結果を仮想モデルに反映するか判断する。PORTA（ ）は、図16の機能関数選択テーブル31にないので、「N」へ進む。

【0370】以上で、図19の処理を終了し、図17の処理に戻る。

【0371】図17のステップS411（システムI/Fモデルシミュレーション）で、システムI/Fモデル8は、HIP入力レジスタ310からシミュレーション結果を読み出して、端子信号出力（0, 1, 0, 0）即ち「#04H」を、ステップS410（シミュレータ間

I/F処理）を介してバスI/Fモデル6に渡す。

【0372】ステップS403で、バスI/Fモデル6は、「#04H」を受領し、機能関数I/Fモデル5に渡し、更にCPUモデル4に渡す。

05 【0373】図9のステップS109で、CPUモデル4は、機能関数I/Fモデル5より受領した「#04H」をレジスタAに格納する。

【0374】このように、第2実施例によれば、第1実施例と同様、仮想モデルを使用することで、ハードウェアモデルシミュレーションに要する時間を大幅に短縮できるとともに、1つの周辺回路のシミュレーション結果が、他の周辺回路のシミュレーション結果に影響を及ぼす場合であっても、仮想モデルのシミュレーション結果をハードウェアモデルに反映したり、ハードウェアモデルのシミュレーション結果を仮想モデルに反映させることで、シミュレーション結果に誤検証を生ずることなく、正確なシミュレーションができるようになる。

【0375】また、第1実施例のように、仮想モデルを使用できる条件として、1つの周辺回路のシミュレーション結果が、他の周辺回路のシミュレーション結果に影響を及ぼさないという制約がなくなるので、仮想モデルとして登録できるサブモデルの種類を第1実施例に比べて増やすことができる。

【0376】また、ハードウェアシミュレータを使用する対象を、仮想関数選択テーブルの内容を書き替えるだけで、簡単に選択することができる。この結果、問題の残ったハードウェアモデルだけを集中して検証することができるようになり、短期間でターゲットプログラムや周辺回路の問題を解決することができる。例えば、キー入力モデルのいずれかに異常があるが、出力ポート側は正常に動作していることが確かめられれば、出力ポート側を仮想モデルに登録することで、デコード出力までのシミュレーション時間を短縮し、入力ポート側を重点的に検証することができる。

35 【0377】[第3実施例] 第1実施例および第2実施例では、ハードウェアシミュレーションはマイコンシミュレータからの周辺回路処理ルーチンとして位置付けられ、IORD, IOWR等入出力に関する機能関数により呼出されていた。しかしながら、例えば通信回線からのデータ受信等外部要因による事象を受け入れる電子機器もあり得る。第3実施例は、このような電子機器に対して有効なシステムシミュレーションを実行できるシステムシミュレータおよびシステムシミュレーション方法を提供するものである。すなわち、ハードウェアモデルの外部端子を監視し、状況変化をキャッチするとハードウェアシミュレーションを起動する。このときのハードウェアシミュレーションは周辺回路処理サブルーチンにおけるハードウェアシミュレーションに対する割り込みとして優先的に扱われる。ハードウェアシミュレーションの結果は、必要に応じて仮想モデルの更新に反映さ

れ、以後仮想モデルシミュレーションによって代替できるような構成とすることにより、システムシミュレーションの高速化を図るのである。

【0378】図22は、本発明のシステムシミュレータの第3実施例の機能ブロック図である。図22と図13を対比すれば、第3実施例は第2実施例におけるハードウェアシミュレータ2-1に割込I/Fモデル24とバス要求I/Fモデル25とを付加した点が異なっているのがわかる。また、シミュレーション装置のSIO端子23およびシリアル入出力I/FモデルS__SIO(図2)を顕示した。割込I/Fモデル24とバス要求I/Fモデル25はCPUモデル42と接続されてシミュレータ間I/Fモデルを構成するが、それらの接続端子信号はターゲットプログラムに基づく信号変化とは無関係に変化するため、バスI/Fモデル6およびシステムI/Fモデル8とは別個に設けたものである。

【0379】割込I/Fモデル24は、ハードウェアモデル22からの割込み要求信号をCPUモデル42に伝えたり、CPUモデル42からハードウェアモデル22側に割込み許可信号を伝えたりする。

【0380】バス要求I/Fモデル25は、ハードウェアモデル22からのバス開放要求信号をCPUモデル42に伝えたり、CPUモデル42からハードウェアモデル22側にバス使用許可信号を伝える。

【0381】シリアル入出力I/FモデルS__SIOは、SIO端子23から入力される信号を復調して仮想モデル72あるいは所定の記憶領域S__RAM(図2)に渡し、また仮想モデル72あるいは所定の記憶領域S__RAMから入力される信号を変調してSIO端子23に出力する。

【0382】第3実施例においても、図2に示したシミュレーション装置のブロック図およびその説明は第1実施例および第2実施例と同様に適用できる。さらに、これまでの実施例では使用されることがなかった割込制御装置S__INT、DMA装置S__DMAおよびシリアル入出力I/FモデルS__SIOが有用になる。

【0383】図23は第3実施例におけるハードウェアモデル22の構成図であり、図14に示した第2実施例におけるハードウェアモデル21に対して、HDA設定レジスタH306付きDMAモデルH305、およびHSI設定レジスタH308付きSIOモデルH307が付加されている。DMAモデルH305、SIOモデルH307は、システムI/Fモデル8と接続されるとともに、それぞれバス要求I/Fモデル25、割込I/Fモデル24と接続される。また、HDA設定レジスタH306、HSI設定レジスタH308は、データ更新I/Fモデル9と接続され、それぞれDMAモデルH305、SIOモデルH307の更新のために使用されるとともに、仮想モデル72の更新のために出力される。

【0384】SIOモデルH307は、SIO端子23

と接続され、SIO端子23は図示しない通信端末と接続され、通信データを授受する。SIOモデルH307は、通信端末から1バイト分のデータを受信すると、割込み要求信号を割込みI/Fモデル24を介してCPUモデル42に出力する。さらに、受信データをデータメモリモデルH303にDMA転送するため、DMAモデルH305にデータ転送要求信号DRQを出力する。DMAモデルH305にデータ転送要求信号DRQが受け付けられると、SIOモデルH307は、1バイトの受信データをDMAモデルH305に出力する。これらの処理はいずれも、入力端子に所定の端子信号を与え、接続情報に基づき各接続点および出力端子の信号レベルを演算により求めることにより行われる。

【0385】HSI設定レジスタH308には、CPUモデル42のストア命令により通信速度やパリティなどのパラメータを設定することができる。

【0386】DMAモデルH305は、CPUモデル42を介さずにデータ転送を可能にするもので、データメモリモデルH303の所定の領域の記憶内容を他の領域に転送したり、SIOモデルH307との間でデータ転送を行う。本実施例では、SIOモデルH307で受信したデータをデータメモリモデルH303に転送する場合を例に説明する。

【0387】DMAモデルH305は、SIOモデルH307からデータ転送要求信号DRQを受け付けると、CPUモデル42にバス開放要求信号をバス要求I/Fモデル25を介してCPUモデル42に出力する。CPUモデル42は、バス開放要求信号HRQを受け付けると、バス使用許可信号HLDAをバス要求I/Fモデル25を介してDMAモデルH305に返すとともに、バスを開放する。DMAモデルH305は、バス使用許可信号HLDAを確認すると、SIOモデルH307にデータ転送許可信号DACKを返す。SIOモデルH307は、データ転送許可信号DACKを確認すると、1バイトの受信データをDMAモデルH305に出力する。DMAモデルH305は受信データを一旦DHA設定レジスタH306に格納した後、データメモリモデルH303の所定のアドレスに受信データを書き込む。以上の処理を繰り返し、DMA転送が終了すると、DMAモデルH305は、バス開放要求信号を非活性化してバス要求I/Fモデル25を介してCPUモデル42に出力する。CPUモデル42は、非活性化したバス開放要求信号HRQを確認すると、非活性化したバス使用許可信号HLDAをバス要求I/Fモデル25を介してDMAモデルH305に返すとともに、バス使用権を復活する。以上の処理は接続方法と端子信号をもとに演算されて、所定の結果を得る。

【0388】図24は、第3実施例における仮想モデル72の構成図であり、図15に示した第2実施例における仮想モデル71に対して、VRTテーブルV406付

き仮想通信モデルV405が付加されている点が異なる。

【0389】仮想通信モデルV405は、図23におけるDMAモデルH305およびSIOモデルH307に対応し、またVRTテーブルV406は図23におけるHDA設定レジスタH306およびHSI設定レジスタH308に対応する。

【0390】仮想通信モデルV405は、SIO端子23を介して受信した信号を復調して、データメモリモデルのHDMテーブルH304またはVDMテーブルV404に記憶させたり、データメモリモデルのHDMテーブルH304またはVDMテーブルV404の記憶内容を変調してSIO端子23を介して送信できればよい。例えば、通信相手側の変調する前の文字データを直接仮想データメモリV403に書き込むようにしてもよいし、あるいは、別に構成したモデム(MODEM)で受信・復調した文字データを図2に示すS_DMAを利用して仮想データメモリV403にDMA転送するようにしてもよい。本実施例では、シミュレーション装置が有するS_SIOとS_DMAを使用して、所定の記憶領域HDMテーブルH304またはVDMテーブルV404に転送する例を説明する。

【0391】VRTテーブルV406には、SIOモデルの通信速度や通信モード、パリティなどの通信パラメータと、DMAの転送元アドレスや転送先アドレスと、割り込み制御装置S_INTの割り込みベクタなどを設定する。

【0392】シリアル入出力I/F装置S_SIOは、SIO端子23と接続され、通信相手側と通信データを授受する。また、シリアル入出力I/F装置S_SIOは、CPUモデル42のストア命令により仮想通信モデルV405を介して、通信速度や通信モード、パリティなどが設定される。シリアル入出力I/F装置S_SIOが1バイト分の通信データを受信すると、シリアル入出力I/F装置S_SIOは割り込み制御装置S_INT(図2)に割り込み要求信号を出力する。割り込み処理が起動されると、DMA装置S_DMAは、シリアル入出力I/F装置S_SIOから通信データを読み込んで、メモリS_RAM内に形成された所定の記憶領域VDMテーブルV404にDMA転送を開始する。

【0393】DMA転送が終了すると、VDMテーブルV404の内容は、ハードウェアモデル22に渡され、データ更新処理によりHDMテーブルH304の内容が更新される。

【0394】第2実施例では、図14と図15とを見比べれば明らかなように、ハードウェアモデルを構成する各モデルと、仮想モデルを構成する各モデルとは1:1対応の関係にあるが、第1、第3実施例では、上述のように2:1対応の関係にある。即ち、ハードウェアモデルを構成する複数のサブモデルをまとめた機能を1つの

仮想モデルで実現することができる。

【0395】図25は、機能関数選択テーブル32の内容例であり、第1実施例および第2実施例における機能関数選択テーブル3、31の内容(図6、図16)と対05
比すると、フラグが設けられている点に特徴がある。シミュレータ操作者が、図示しない編集装置を使って、このフラグを書き替えることができる。フラグが「1」であって引数の条件に合えば、仮想モデルを選択することができ、「0」であれば、仮想モデルを選択することができない。

【0396】このように、フラグを書き替えるだけで、仮想モデルを使うか否かを決定することができるので、第1や第2実施例のように機能関数選択テーブル全体を書き替える場合に比べ、ハードウェアモデルと仮想モデルの選択の設定が容易にできるようになる。

【0397】図25(A)は、ハードウェアモデル22でSIO、DMA転送を行う場合、図25(B)は、仮想モデル72で受信したシリアルデータを受信し、ハードウェアモデル22のデータを更新する場合の機能関数20
選択テーブル32の内容例を示す。

【0398】ここで、第3実施例における、CPUモデルシミュレーション、周辺回路処理サブルーチンおよびハードウェアモデルシミュレーションの特徴部分について、図27、図28および図29により説明し、以下の動作説明の流れを円滑にする。

【0399】まず、図27のCPUモデルシミュレーションのフローチャートを参照すると、本フローチャートは、図9のフローチャートに、割り込み要求の有無をチェックするステップ(S361)、バス開放要求の有無をチェックするステップ(S362)、割り込みに関するステップ(S367、S368)およびバス開放に関するステップ(S363~S366)が付加されている。

【0400】割り込み要求の有無やバス開放要求の有無は、CPUモデル42が割り込み端子、バス要求端子に割り込み要求信号や、バス開放要求信号が入力されているか35
否かを確認することにより検知する。本例では、説明を簡単にするために、スタート直後に、このような確認するステップS361、S362を設けたが、実際には、CPUモデルシミュレーションの処理とは独立して、割り込み的に要求を受け付けるようにしている。

【0401】ステップS361(割り込み要求有?)で、ハードウェアモデル22のいずれかのサブモデルが割り込み要求信号を発生し、割り込みI/Fモデル24を介してCPUモデル42に入力されたか否かを判断する。割り込み要求信号が有効であれば「Y」に進み、ステップ367の処理に移る。割り込み要求信号が有効でなければ「N」に進み、ステップ362の処理に移る。

【0402】ステップS367(割り込み受付可?)で、CPUモデル42は、割り込み要求が受付可能かどうかチェックし、受付可能であれば、「Y」へ進み、ステップ40
50

S368で所定の割込み処理を行う。本例では、説明を簡単にするために、SIOモデルH307だけから割込みがかかり、DMAモデルH305を起動する例を説明するが、実際には、割込み要求は多くの周辺回路から発せられるので、割込みコントローラ(図2の割込制御装置S__INTに相当)を設けて、割込み要求の調停をすることもある。割込みコントローラは、CPUモデル42に対して割込み要求と、優先順位の高い方の割込みベクタ(どこから割込み要求があって、どの割込み処理をするかを指示するもの)を出力する。CPUモデル42が割込みベクタに対応する割込み処理を実行する。割込み処理は、割込みベクタに対応する割込み処理が記述されたアドレスをプログラムカウンタPCにセットして、図27の流れ図を呼び出すことで、再帰的に処理される。

【0403】ステップS362(バス開放要求有?)で、バス開放要求信号(HRQ:ホールドリクエスト)が入力された場合、CPUモデル42はそれが受付可能かどうかをチェックし、受付可能であれば、「Y」に進み、ステップS363の処理に移る。受付不可能であれば、「N」に進み、ステップS101の処理に移る。

【0404】ステップS364(バス使用許可)で、CPUモデル42は、以下のようにバスを開放する処理を行う。本例では、説明を簡単にするために、DMAモデルH305だけからバス要求がかかり、DMAモデルH305がバスを占有する例を説明するが、実際には、バス要求は多くの周辺回路から発せられるので、バスアービタ(不図示)を設けて、バス要求の調停をすることもある。

【0405】CPUモデル42は、バス要求を受け付けると、バス使用許可信号(HLDA:ホールドACK)をバス要求I/Fモデルを介してDMAモデルH305に返す。

【0406】ステップS365(バス開放)で、CPUモデル42は、バスを開放して、バス使用权をDMAモデルH305に渡す。通常、CPUモデル42がバスを開放すると、CPUモデルは何も処理できなくなる。

【0407】ステップS366(バス開放?)で、CPUモデル42は、バス開放要求信号が非活性化するのを待つ。即ち、DMAモデルH305がDMA処理中で、バス開放要求信号が活性化している間は、「N」に進み、非活性化したことを検出すると、「Y」に進み、ステップS110に移る。

【0408】ステップS110(ブレイク?)で、処理停止または中断の命令でなければ、CPUモデル42は「N」へ進み、ステップS361~S109の処理を繰り返す。処理停止または中断の命令であれば、CPUモデル42は「Y」へ進み、処理を中断または終了する。

【0409】図28は、第3実施例における周辺回路処

理サブルーチンのフローチャートを示し、図29は、図28のステップS413(CPU特殊端子入出力)の詳細フローチャートを示すものである。

【0410】次に、図28と図29を参照すると、本フローチャートは、図17のシステムI/FモデルシミュレーションS411を詳細化するとともに、図17のフローチャートに、「外部端子変化?」(S216), 「割込み要求有り?」(S432), 「割込み要求信号出力」(S433), 「バス要求有?」(S417), 「バスホールド出力」(S418)および「ホールドアクノリッジ」(S419)を追加したことがわかる。

【0411】ステップS431(仮想モデル呼出し?)で、マイコンシミュレータ1-2(図22)は、機能関数選択テーブル32を参照して、仮想モデルシミュレータ72、またはハードウェアシミュレータ2-2のいずれを使用するかを判断する。第1、第2実施例では、機能関数選択テーブルに検証すべき機能関数が登録されているか、登録されていれば引数条件を満たしているかを判断して、いずれかのシミュレータ1-2, 2-2を選択していた。本実施例においては、機能関数選択テーブル32内にフラグを設け、フラグの有無で、シミュレータ1-2, 2-2のいずれを選択するか判断するようにした。

【0412】具体的には、マイコンシミュレータ1-2は、検証すべき機能関数のフラグが「1」で、引数条件を満たしていれば、「Y」へ進み、仮想モデルシミュレータ72を使用し、ステップS422(仮想モデルシミュレーション)を実行する。逆に、フラグが「0」であれば無条件に「N」に進み、ハードウェアシミュレータ2-2は、ステップ412(ハードウェアモデルシミュレーション)を実行すべく処理を進める。即ち、ステップS403(バスI/Fモデルシミュレーション)で、バスI/Fモデル6(図22)は、機能関数I/Fモデル5から機能関数を受け取り、端子信号に変換し、ステップS410(シミュレータ間I/Fモデルシミュレーション)でシミュレータ間コード変換を行い、システムI/Fモデル8に端子信号を供給する。ステップS411(システムI/Fモデルシミュレーション)で、システムI/Fモデル8は、端子信号を受け取り、これをハードウェアモデル22に渡す。

【0413】ステップS212~S215およびステップS216がI/FモデルシミュレーションS411を構成する。ステップS212~S215は、第1実施例の図10と同様に動作する。

【0414】ステップS216(外部端子変化?)で、ハードウェアモデル22は、ハードウェアモデル22に接続された外部端子(SIO端子23など)の状態を定期的に確認し、変化があれば、ハードウェアモデルシミュレーションS412内の所定のサブモデルシミュレーションを起動する。例えば、ハードウェアモデル22

(図23)は、SIOモデルH307のSIO端子23からの入力信号の状態を定期的に確認し、変化があれば、SIOモデルH307のシミュレーションを起動する。

【0415】なお、図28において、端子信号変化(S213)を先に検出し、「システムI/F完了？」(S215)で全端子信号の受信が完了しない間は、CPUモデル42からの信号受信を優先させ、「外部端子変化？」(S216)の処理をマスクするようにしてもよい。

【0416】このように、第2実施例では、CPUモデル42から発せられる端子信号だけに依じてハードウェアシミュレータ2-1を起動するようにしていたのに対し、本例では、このように、外部端子が変化したかをチェックし、変化したらハードウェアシミュレーションを開始するようにした。

【0417】ステップS412(ハードウェアモデルシミュレーション)の結果、割込要求信号やバス要求信号などCPU特殊端子信号が活性化した場合には、ハードウェアシミュレータ2-2は、ステップS413(CPU特殊端子入出力)を処理する。非活性のままであれば、ステップS424、S425で仮想モデルデータ更新の処理に移る。

【0418】ステップS413の具体的な処理の流れを図29を参照して説明する。

【0419】ステップS432(割込み要求信号有り?)で、ハードウェアモデルシミュレーション(S412)の結果、ハードウェアモデル22(図22)は、割込要求信号IRQが活性化したかどうかを判断する。IRQが活性化していれば、「Y」へ進み、ステップS423の処理に移り、IRQが非活性のままであれば、「N」へ進み、ステップS417の処理に移る。

【0420】ステップS423で、ハードウェアモデル22内のSIOモデルH307(図23)は、割込みI/Fモデル24経由でCPUモデル42に対して割込み要求信号IRQを出力する。

【0421】ステップS417(バス要求有り?)で、ハードウェアモデル22(図22)は、ハードウェアモデルシミュレーション(S412)の結果、バス要求信号HRQが活性化したかどうかを判断する。HRQが活性化していれば、「Y」へ進み、ステップS418の処理に移り、HRQが非活性のままであれば、「N」へ進み、ステップS424の処理に移る。

【0422】ステップS418(バス開放要求出力)で、ハードウェアモデル22内のDMAモデルH305(図23)は、バス要求I/Fモデル25経由でCPUモデル42に対してバス開放要求信号HRQを出力する。

【0423】図27のステップS362で、CPUモデル42は、バス開放要求信号HRQが活性化したことを

検出すると、S363→S364→S365と進み、バス使用権を放棄するとともに、バス使用許可信号HLDAを活性化して、バス要求I/Fモデル25経由でDMAモデルH305に返す。

05 【0424】ステップS419で、DMAモデルH305(図23)は、バス要求I/Fモデル25経由でCPUモデル42からバス使用許可信号HLDAを受領したか否かを判断する。HLDAを受領しなければ、「N」へ進み、ステップS419に留まる。HLDAを受領すれば、バス使用権を取得して「Y」へ進み、ステップS412に戻り、DMAモデルH305のシミュレーションを開始する。

【0425】ステップS412で、DMAモデルH305のシミュレーションが終了すると、DMAモデルH305は、バス開放要求信号HRQを非活性化した信号を出力するとともに、バスを開放する。以後、S432→S417→S418と進む。

【0426】ステップS418(バス開放要求出力)で、ハードウェアモデル22内のDMAモデルH305(図23)は、バス要求I/Fモデル25経由でCPUモデル42に対して、非活性化したバス開放要求信号HRQを出力する。

【0427】図27のステップS366で、CPUモデル42は、バス開放要求信号HRQが非活性化したことを検出すると、バス使用権を取得するとともに、バス使用許可信号HLDAを非活性化して、バス要求I/Fモデル25経由でDMAモデルH305に返す。

【0428】図29のステップS419で、DMAモデルH305(図23)は、バス要求I/Fモデル25経由でCPUモデル42から、非活性化したバス使用許可信号HLDAを受領したか否かを判断する。HLDAを受領しなければ、「N」へ進み、ステップS419に留まる。非活性化したHLDAを受領すれば、「Y」へ進み、ステップS412に戻る。

35 【0429】ステップS432、S417でいずれも「N」であったとき、ハードウェアモデル22は、ステップS424に進む。

【0430】ステップS424(仮想モデルデータ更新要?)で、ハードウェアモデル22は、ハードウェアモデルシミュレーション(S412)の結果が仮想モデル72に関係しているか否かを機能関数選択テーブル32のフラグを参照して判断する。ハードウェアモデル22は、ハードウェアモデルシミュレーション(S412)に対応する機能関数が機能関数選択テーブル32に登録されている、かつその引数条件を満たし、フラグが「1」であれば、「Y」へ進み、ステップS425の処理に移る。対応する機能関数のフラグが「0」であれば、引数条件に関係なく「N」へ進み、ステップS212の処理に移る。

50 【0431】ステップS425、S421の処理は第2

実施例と同じであり、説明を省略する。

【0432】ステップS403、S420、S421は、第2実施例の図17と同じであり、また、ステップS205～S208は、第1実施例の図10と同じであり、説明を省略する。

【0433】次に、第3実施例の動作を、図26に示すターゲットプログラムを例にとり、図27～図33のフローチャートおよび図34のタイミングチャートを用いて説明する。以下の説明は、重複を回避するため、第1実施例および第2実施例について行ったようなプログラム1行ごとに詳述することはせず、より要約的なものに止める。

【0434】本プログラムは、SIO端子23で受信したデータをRAMにDMA転送するというものである。第1のケースとして、ハードウェアシミュレータ2-2によって行う場合と、第2のケースとして、仮想モデル72を用いて行う場合とについて述べる。

【0435】[第1のケース：ハードウェアシミュレータ2-2]

《ターゲットプログラムT301：SIO初期設定の処理》図26において、先ずCPUモデル42は、ハードウェアモデル22のSIOモデルH307（図23）に通信速度、パリティ有無、受信モード等を設定する。この設定は、先の実施例と同様に、アドレスとしてSIOモデルH307を指定したMOV命令（図8）により行えばよい。

【0436】《ターゲットプログラムT302：DMA初期設定の処理》CPUモデル42は、同様にして、DMAモデルH305（図23）に、使用するチャンネル番号、転送モード、アドレス増／減の別や、DMA転送元およびDMA転送先（データメモリモデル）の各開始アドレスと終了アドレス等を設定する。

【0437】《ターゲットプログラムT303：SIO割込み許可》CPUモデル42は、SIOモデルH307に割込みの許可を与える。SIOモデルH307（図23）は、SIO端子23に入力されるスタートビットが変化したか否かを検出する処理を開始する。

【0438】以上の処理T301、T302およびT303は、図27において、S361とS362では「N」であるから、ステップS101～S110を3回繰返すことにより実行される。そして、ステップS123およびS128では、図28の周辺回路処理サブルーチンが起動されるが、ステップS216（外部端子変化？）、S432（割込み要求有り？）およびS417（バス要求有り？）では、いずれも「N」であるから、図17と同一の処理ルートを進めることになる。

【0439】《ターゲットプログラムT304：割込み発生》CPUモデル42は、T303以降の図示しない処理を継続して実行する。

【0440】ここで、SIO端子23に調歩同期式によ

り通信データが入力されたとする。

【0441】時刻t0（図30）のステップS216（図28の外部端子変化？）で、SIOモデルH307は、SIO端子23が「1」から「0」に変化するとスタートビットであると判断する。SIOモデルH307は、このスタートビットを検出すると（図30（b））、「Y」へ進み、ハードウェアモデルシミュレーション（S412）を起動する。ここで、時刻t1～t9のステップS412で、SIOモデルH307は、SIO端子23より1ビットずつ通信データを受信する。ここで、受信データ（D7～D0）は、「01000001」、すなわち「41H」であり、文字コード「A」に相当する。

【0442】時刻t10のステップS412で、SIOモデルH307は、1バイト（8ビット）分のデータを受信し、ストップビットを検出すると、割込I/Fモデル24に割込み要求信号IRQを出力するとともに（図30（d））、DMAモデルH305（図23）にデータ転送要求信号DRQを出力する（図30（e））。

【0443】ステップS432（割込み要求有り？）で、ハードウェアモデル22は、この割込み要求を検知すると、ステップS433で割込I/Fモデル24を介してCPUモデル42に割込み要求信号IRQを出力する。

【0444】ステップS361（図27、割込み要求有り？）で、CPUモデル42は、割込み要求信号IRQを検出すると「Y」へ進む。

【0445】ステップS367（割込み受付可？）で、CPUモデル42は、割込み受付が可能かを確認する。今回の割込み要求より優先順位の高い割込み処理を実行中であれば、受付不可として「N」へ進む。ここでは、割込み要求の受付が可能であるとして、「Y」へ進む。

【0446】ステップS368（割込み処理）で、CPUモデル42は、定められた割込み処理ルーチン呼び出す。ここで、割込み処理ルーチンは、DMAモデルH305に対して、チャンネル0のDMAを許可する命令であり、アドレス1900H番地以降に記述してあるとする。

【0447】CPUモデル42は、プログラムカウンタに1900Hを格納して、図27に示すCPUモデルシミュレーションを再帰的に呼び出す。ステップS361→S362→S101→S102→S123→S104→S105→S106→S107→S128と処理が進む。

【0448】時刻t11～t14の周辺回路処理サブルーチン（図28）において、ステップS431→S403→S410→S411→S412の順に処理が行われ、1900H番地の命令に対応する端子信号I/O、/WR、/CS0、ADRS、DATAなどがハードウェアモデル22に供給される（図30（h）～

(o))。

【0449】時刻t14のステップ412で、DMAモデルH305は、DMA転送が可能になる。以上で、1900H番地の処理が終了し、ステップS413→S424→S411と進み、ハードウェアシミュレーション完了信号「DONE」を、S410→S403→S109 (図27)の順にCPUモデル42に返す。これにより、CPUモデル42は、開始に戻り、次のシミュレーション待ちとなる。

【0450】一方、ハードウェアシミュレータ2-2は、ステップS213→S216と進む。

【0451】時刻t15のステップS216 (外部端子変化?)で、ハードウェアシミュレータ2-2は、SIOモデルH307 (図23)からのデータ転送要求信号DRQを検知すると、ハードウェアモデルシミュレーション (S412)のDMAモデルH305処理を起動し、DMAモデルH305は、バス解放要求信号(HRQ)を活性化 (図30(f))。以後、ステップS432→S417→S418 (図29)と進む。

【0452】時刻t15のステップS418 (バス開放要求)で、DMAモデルH305は、バス要求I/Fモデル25を介してCPUモデル42にバス解放要求信号HRQを出力して、ステップS419に移り、CPUモデル42からバス使用許可信号HLDAが活性化するのを待つ。

【0453】時刻t15のステップS362 (図27、バス解放要求有?)で、CPUモデル42は、HRQ信号が活性化していることを検出すると、「Y」へ進む。

【0454】ステップS363 (バス解放受付可?)で、CPUモデル42は、今回のバス解放要求より優先順位の高い処理を実行中であれば、受付不可として「N」へ進む。ここでは、バス解放要求の受付が可能であるとして、「Y」へ進む。

【0455】時刻t16のステップS364 (バス使用許可)で、CPUモデル42は、バス使用許可信号(HLDA: ホールドアクリッジ)をバス要求I/Fモデル25経由でDMAモデルH305に供給する (図30(g))。

【0456】ステップS365 (バス解放)で、CPUモデル42は、バスの解放を行い、ステップS366 (バス開放)で、DMAモデルH305がバスを開放するのを待つ。

【0457】一方、ステップS419 (バス使用許可?)で、DMAモデルH305は、バス使用許可信号HLDAを検知すると、「Y」へ進み、ステップS412に移る。

【0458】時刻t17のステップ412で、DMAモデルH305は、バス使用権を取得し、DMA転送処理を開始する。

【0459】まず、DMAモデルH305は、端子信号

I/O、/CS1、ADRS (SIO)がSIOモデルH307に供給される (図30(h)(1)(n))。

【0460】時刻t18で、DMAモデルH305は、端子信号/RDを活性化し (図30(j))、SIOモデルH307よりDATA「A」を読み込む (図30(o))。また、SIOモデルH307は、データの読み出しを検知すると、データ転送要求信号を非活性化する (図30(e))。

【0461】時刻t19、t20で、DMAモデルH305は、端子信号を元に戻し、SIOモデルH307からの読み出しサイクルを終了する。

【0462】時刻t21で、DMAモデルH305は、端子信号/MEM、/CS2、ADRS (2200H)をデータメモリモデルH303に供給する (図30(h)(m)(n))。

【0463】時刻t22で、DMAモデルH305は、端子信号/WRを活性化し (図30(j))、データメモリモデルH302の2200H番地にDATA「A」を書き込む (図30(o))。

【0464】時刻t23、t24で、DMAモデルH305は、端子信号を元に戻し、データメモリモデルH303への書き込みサイクルを終了する。

【0465】このように、時刻t25~t35の間もSIO→DMA→メモリの順にデータが転送され、通信データが終了するまで、繰り返される。

【0466】今の場合、SIOモデルH307は、受信したデータ「A」、「B」を読み込み、このデータをデータメモリモデルH303に順次DMA転送する。DMAモデルH305は、HDMAテーブルH304の2200、2201番地にそれぞれ「A」、「B」を記憶する。

【0467】時刻t36のステップS412 (図29)で、DMAモデルH305は、DMA転送が終了すると、バス開放要求信号HRQを非活性化する。以降、ステップS432→S417→S418 (図29)と進む。

【0468】ステップS418 (バス開放要求出力)で、DMAモデルH305は、バス解放要求信号(HRQ)を非活性化 (図30(f))。DMAモデルH305は、バス要求I/Fモデル25を介してCPUモデル42にバス解放要求信号HRQを出力して、ステップS419に移り、CPUモデル42からバス使用許可信号HLDAが非活性化するのを待つ。

【0469】一方、時刻t37のステップS366 (図27、バス解放?)で、CPUモデル42は、HRQ信号が非活性化していることを検出すると、バス使用許可信号HLDAを非活性化して、バス要求I/Fモデル25経由でDMAモデルH305に供給する (図30(g))。その後、CPUモデル42は、ステップS10で「Y」へ進み、開始に戻る。

【0470】一方、ステップS419（バス使用許可？）で、DMAモデルH305は、非活性化したバス使用許可信号HLDAを検知すると、「Y」へ進み、ステップS412に移る。以上で、DMA転送は終了し、ステップS412→S432→S417→S424に進む。

【0471】ステップS424（仮想モデルデータ更新要？）で、ハードウェアモデル2-2は、ハードウェアモデルシミュレーションS412の結果、仮想モデル72側のデータ更新が必要か否かを判断する。ハードウェアモデル2-2は、データメモリモデルH303のHDMテーブルの内容が書き変わったことを検知し、かつ、機能関数選択テーブル32（図25（A））のフラグが「1」であることを確認し、「Y」に進む。

【0472】ステップS425（仮想モデルデータ更新起動）で、ハードウェアモデル2-2は、HDMテーブルH304の2200, 2201番地に記憶した「A」, 「B」を、仮想モデル42に渡す。

【0473】ステップS421（図28）で、仮想モデル72は、受領した更新データを仮想データメモリモデルV403に付属するVDMテーブルV404の2200, 2201番地に書き込む。

【0474】ステップS212で、ハードウェアシミュレータ2-2は、DMAモデルH305のDMA転送処理が完了したことを示す「DONE」をステップS410→S403→S128経由でCPUモデル42に返す。

【0475】この後、ハードウェアシミュレータ2-2は、ステップS411でアイドル状態、即ち、ステップS213（端子信号変化？）またはS216（外部端子変化）のいずれかが「Y」になるのを待つ。

【0476】《ターゲットプログラムT311：データメモリモデル読み出し》CPUモデル42は、仮想モデル72の仮想データメモリモデルV403（図24）の2200番地からデータ“A”を読み出す。

【0477】《ターゲットプログラムT312：データ判断》CPUモデル42は、読み出したデータが「A」（41Hとする）であるか否かを判断する。「A」であれば、「Y」へ進み、T313の処理に移る。「A」でなければ、「N」へ進み、T314の処理に移る。

【0478】《ターゲットプログラムT313：ポート出力「1」》CPUモデル42は、読み出したデータが「A」であれば、出力ポートに「1」を出力し、正確に受信できたことを表示する。

【0479】《ターゲットプログラムT314：ポート出力「0」》CPUモデル42は、読み出したデータが「A」以外であれば、出力ポートに「0」を出力し、正確に受信できなかったことを表示する。

【0480】以上で、図26のターゲットプログラムによる動作説明を終える。

【0481】このように、ハードウェアシミュレータ2-2を使ってDMA転送を行い、データメモリモデルH303の記憶内容を書き替えても、データ更新機能により、仮想モデル72側の記憶内容も同時に更新されるので、その後のターゲットプログラムで、仮想モデル72側の仮想データメモリモデルV403から記憶内容を読み出しても、正しい処理が実行できる。

【0482】つまり、DMA転送などCPUモデル42が関知しない状態で、ハードウェアモデル22のデータメモリモデルH303が書き替えられても、仮想モデル72側の仮想データメモリモデルV403も同時に更新するようにしたので、DMA転送直後のT311（図26）で仮想データメモリモデルV403を読み出しても、ターゲットプログラムT312で判断を誤ることなく正常に処理される。

【0483】また、シミュレーションの処理を仮想モデル72で行うか、ハードウェアシミュレータ2-2で行うかを判断テーブル内に設けたフラグにより切り替えるようにした。このため、シミュレータ装置の操作者は、テーブルの内容（機能関数や判断条件）を書き換えることなく、フラグの書き換えだけで仮想モデル7とハードウェアモデル22の選択が容易に行える。従って、操作者は、システムシミュレーションの最中に、シミュレーションの用途に応じて適宜フラグを書き替え、問題のあるハードウェアモデル22だけを集中して検証することができる。問題が解決したハードウェアモデル22内のサブモデルは、仮想モデル72に置き換えてシミュレーションすることができるので、検証に要する時間を大幅に短縮できる。

【0484】さらに、外部端子の変化をチェックする手段を設けたので、CPUモデル42の端子信号の変化だけでなく、外部端子の信号変化をトリガとして、ハードウェアシミュレータ2-2を起動することができる。この結果、SIOのような通信モデルがシミュレーションできるようになった。

【0485】また、ハードウェアシミュレータ2-2側に、割込みI/Fモデル24やバス要求I/Fモデル25を設けたので、マイコンシミュレータ1側の動作状態に関わらず、要求信号を渡すことができる。その結果、DMAやSIOのシミュレーションが可能になった。

【0486】[第2のケース：仮想モデルシミュレータ72によるDMA転送の動作説明]次に、仮想モデルの仮想通信モデルV405を使って、SIOデータを受信する例を説明する。この動作の概要は次のようである。

すなわち、仮想通信モデルV405は、シリアル入出力S_SIOから受信データを得る。このデータは仮想データメモリモデルV403のデータテーブルVDMテーブルV404に書き込まれる。これと同時に、ハードウェアモデル22のデータメモリモデルH303のデータテーブルHDMテーブルH304のデータも更新され

る。次に、ハードウェアモデル22のデータメモリモデルH303のデータテーブルHDMテーブルH304からデータを読み出す。

【0487】以下、図26のターゲットプログラムに沿って説明する。

【0488】《ターゲットプログラムT301：SIO初期設定の処理》CPUモデル42は、仮想通信モデルV405内のVRTテーブルV406に諸条件を設定する。設定内容はハードウェアシミュレータ2-2による場合と同じである。この設定は、仮想通信モデルV405に記述されたプログラムに従って、シミュレーション装置のシリアル入出力I/FモデルS_SIO（図2）に対して行われ、その後、データ更新（図28、S423）によりSIOモデルH307内のHSI設定レジスタH308にも書き込まれる。

【0489】《ターゲットプログラムT302：DMA初期設定》CPUモデル42は、仮想通信モデルV405内のVRTテーブルV406に諸条件を設定する。この設定内容もハードウェアシミュレータ201による場合と同じである。この設定は、通信モデルV405に記述されたプログラムに従って、シミュレーション装置のDMA装置S_DMA（図2）に対して行われ、その後、データ更新（図28、S423）によりDMAモデルH305内のHSI設定レジスタH306にも書き込まれる。

【0490】《ターゲットプログラムT303：SIO割込み許可》CPUモデル42は、仮想通信モデルV405内のVRTテーブルV406に割込みの許可を設定する。この設定は、通信モデルV405に記述されたプログラムに従って、シミュレーション装置のDMA装置S_DMA（図2）に対して設定されるが、DMAモデルH305には、書き込まれないようにする。これは、仮想モデル72とハードウェアシミュレータ2-2が同時に通信データを受信することがないようにするためである。

【0491】《ターゲットプログラムT304：割込み発生》シリアル入出力I/FモデルS_SIO（図2）は、通信データの最初の1バイトを受信し終わると、割込み制御装置S_INT（図2）に対して、割込み要求を行う。割込み制御装置S_INTは、プロセッサS_CPUに対して割り込み要求を行い、割込み処理が記述されたアドレスを渡す。プロセッサS_CPUは、定められた割込み処理ルーチンを呼び出す。ここで、割込み処理ルーチンは、DMA装置S_DMAに対して、チャンネル0のDMAを許可する命令であるとする。

【0492】《ターゲットプログラムT321：DMA許可》図27で、ステップS361→S362→S101→…→S128と進み、CPUモデル42は、仮想通信モデルV405のVRTテーブルV406（図24）にDMA許可を設定するとともに、仮想通信モデルV4

05は、記述されたプログラムに従って、DMA装置S_DMA（図2）にチャンネル0のDMAを許可する。

【0493】以後のSIOデータ受信とDMA転送の処理は、仮想モデルシミュレーションS422やハードウェアモデルシミュレーションS412（図28）に係わらず行われる。即ち、DMA装置S_DMA（図2）は、シリアル入出力I/FモデルS_SIOから受信データ「A」、「B」を読み出し、仮想データメモリモデルV403内のVDMテーブルV404の所定の領域（2200H、2201番地）に対応するメモリS_RAMの領域にDMA転送する。

【0494】《ターゲットプログラムT322：DAM終了？》ステップS422（図28、仮想モデルシミュレーション）で、仮想通信モデルV405（図24）は、DMA転送が終了したことを検知すると、ステップS420の処理に移る。

【0495】ステップS422（ハードウェアデータ更新起動）で、仮想モデル72は、ハードウェアモデル22に仮想データメモリモデルV403内のVDMテーブルV404の更新データ「2200、A」、「2201、B」を渡す。

【0496】ステップS423（データ更新）で、ハードウェアモデル22は、受領した更新データ「A」、「B」をデータメモリモデルH303内のHDMテーブルH304の2200H、2201H番地に書き込む。

【0497】以上で、図28の周辺回路処理サブルーチンを終了し、図27のステップS128に戻り、S109→S110と進み、開始に戻る。

【0498】以上で、割込み処理ルーチンを終了し、割込みが発生したターゲットプログラムのステップに戻る。

【0499】《ターゲットプログラムT311：データメモリモデル読み出し》CPUモデル42は、ハードウェアモデル22のデータメモリモデルH303（図23）の2200番地からデータ“A”を読み出す。

【0500】《ターゲットプログラムT312：データ判断》CPUモデル42は、読み出したデータが「A」（41Hとする）であるか否かを判断する。「A」であれば、「Y」へ進み、T313の処理に移る。「A」でなければ、「N」へ進み、T314の処理に移る。

【0501】《ターゲットプログラムT313：ポート出力「1」》CPUモデル42は、読み出したデータが「A」であれば、出力ポートに「1」を出力し、正確に受信できたことを表示する。

【0502】《ターゲットプログラムT314：ポート出力「0」》CPUモデル42は、読み出したデータが「A」以外であれば、出力ポートに「0」を出力し、正確に受信できなかったことを表示する。

【0503】以上で、図26のターゲットプログラムの検証を終了する。

【0504】ハードウェアシミュレータ2-2によれば、図34のタイミングチャートから明らかなように、SIO端子23に1バイトのデータが入力され、DMA転送によりデータメモリモデルH303に書き込みが終了するまでに30クロック（図30（a）マシンサイクル）程度必要となるが、ハードウェアシミュレータ2-2では、シミュレーションの規模が大きいので、1クロック分を演算するのに、約10mSを要する。したがって、1バイト受信には、300mSもの時間が必要となることになる。このため、SIOモデルH307のデータ転送をハードウェアモデル22では、3bps程度の通信速度はシミュレーションできるが、最近の96Kbps等の高速モデムを、実使用状態でシミュレーションすることは不可能である。

【0505】これに対し、仮想通信モデルV405を使用することで、高速モデムであっても、リアルタイムにシミュレーションすることが可能になる。

【0506】また、仮想通信モデルV405で受信したデータは、ハードウェアモデル22にも書き込まれるので、ターゲットプログラムのT311、T312（図26）をハードウェアモデル22で実行しても、正しい処理が実行される。

【0507】ハードウェアシミュレータ201で、1度、SIOモデルH307やDMAモデルH305が正しく動作することが確認できれば、その後はハードウェアシミュレーションをする必要はなく、これ以降は、機能関数選択テーブル（図25）のフラグを「0」→

「1」に書き替えるだけで、仮想通信モデルV405を使ってターゲットプログラムを検証することができるので、上述の効果をを得ることができるようになる。

【0508】なお、第1～第3実施例では、仮想モデルをマイコンシミュレータの一部として説明したが、これに限定されるものではなく、ハードウェアシミュレータ側に設けてもよいし、マイコンシミュレータやハードウェアシミュレータとは別に構成してもよい。

【0509】また、仮想モデルやハードウェアモデルを構成する周辺回路、あるいは、機能関数、機能関数選択テーブルなどは、第1～第3実施例に限定されるものではなく、電子機器を構成する周辺回路によって適宜変更しうるものである。

【0510】また、シミュレーションを実行するときに、周辺回路の入出力状態をログファイルに記録しておき、次にシミュレーションするときに、ログファイルの内容を仮想モデルやハードウェアモデルの入出力情報として供給することで、シミュレーションに要する時間を短縮することができる。

【0511】

【発明の効果】本発明によれば、電子機器を構成するハードウェアの検証を、電子機器に組み込むプログラムの検証と同時に実行し、かつハードウェアの検証をマイコ

ンシミュレーションで代替する比重を高めたことにより、検証時間の短縮と、検証内容の充実を実現できるという効果がある。

【0512】より具体的な効果については、各実施例の説明の末尾に記載したので重複を回避するため省略する。

【図面の簡単な説明】

【図1】本発明の第1実施例のシステムシミュレータの機能ブロック図

【図2】本発明のシステムシミュレーションの母体装置たるシミュレーション装置のブロック図

【図3】図1におけるハードウェアモデル20の構成例を示すブロック図

【図4】図3のハードウェアモデル20に対応する仮想モデル7の構成図

【図5】図1における仮想プログラムメモリモデルV71内のテーブル例を示す表

【図6】図1における機能関数選択テーブル3の例を示す表

【図7】図1における機能関数I/Fモデル5における仮想関数変換テーブルとその戻り値を示す表

【図8】本発明の第1実施例および第2実施例におけるターゲットプログラムのリスト例を示す表

【図9】本発明の第1実施例および第2実施例におけるCPUモデルシミュレーションのフローチャート

【図10】図9における周辺回路処理サブルーチンS103、S108のフローチャート

【図11】図10における仮想モデルシミュレーションS202のフローチャート

【図12】図1におけるバスI/Fモデル6がハードウェアシミュレータ2にデータを書き込むときのタイミングチャート

【図13】本発明の第2実施例のシステムシミュレータ2-1の機能ブロック図

【図14】図13におけるハードウェアモデル21の構成図

【図15】図13における仮想モデル71の構成図

【図16】図13における機能関数選択テーブル31の例を示す表

【図17】第2実施例における周辺回路処理サブルーチンの詳細フローチャート。実線は処理の流れ、点線はデータの流れを示す。

【図18】第2実施例における仮想モデルシミュレーションの詳細フローチャート

【図19】図17のうち1点鎖線で示された部分のフローチャート

【図20】図17におけるデータ更新S423のフローチャート

【図21】図17におけるデータ更新S421のフローチャート

【図22】本発明の第3実施例のシステムシミュレータ2-2の機能ブロック図

【図23】図22におけるハードウェアモデル22の構成図

【図24】図22における仮想モデル72の構成図

【図25】図22における機能関数選択テーブル32の例を示す表

【図26】第3実施例におけるターゲットプログラムのフローチャート

【図27】第3実施例におけるCPUモデルシミュレーションのフローチャート

【図28】第3実施例における周辺回路処理サブルーチンS123, S128間のフローチャート

【図29】第3実施例において外部端子変化により起動されるハードウェアシミュレーションのフローチャート

【図30】第3実施例におけるシリアル入出力およびDMA転送時のタイミングチャート

【符号の説明】

- 1 マイコンシミュレータ
- 2, 2-1, 2-2 ハードウェアシミュレータ
- 3, 31, 32 機能関数選択テーブル
- 4 CPUモデル
- 5 機能関数I/Fモデル
- 6 バスI/Fモデル
- 7, 70, 72 仮想モデル
- 8 システムI/Fモデル
- 9 データ更新I/Fモデル
- 10, H301 プログラムメモリモデル
- 11, H303 データメモリモデル
- 12 I/OポートAモデル
- 13 キーマトリクスモデル
- 14 I/OポートBモデル
- 15, H312 デコーダモデル
- 16 I/OポートCモデル
- 17 時計モデル
- 18 GUI
- 19 キー入力モデル
- 20, 21, 22 ハードウェアモデル

- 23 SIO端子
- 24 割込I/Fモデル
- 25 バス要求I/Fモデル
- 181 キー入力GUI
- 182 LED表示GUI
- PC プログラムカウンタ
- S_CPU プロセッサ
- S_RAM メモリ
- S_ROM 読み出し専用メモリ
- S_INT 割込制御装置
- S_DMA DMA制御装置
- S_DSP 表示制御装置
- S_INP 入力部
- S_SIO シリアル入出力装置
- S_HDD ハードディスク
- EWS ワークステーション
- V71, V401 仮想プログラムメモリモデル
- V72 仮想時計モデル
- V73 仮想キー入力モデル
- V74 仮想LED表示モデル
- V402, H302 VPMテーブル
- V403 仮想データメモリモデル
- V404, H304 VDMテーブル
- V405 仮想通信モデル
- V406 VRTテーブル
- V407 仮想入力ポートモデル
- V408 VIP入力レジスタ
- V409 仮想キーデコーダモデル
- V410 VKD出力レジスタ
- H305 DMAモデル
- H306 HDA設定レジスタ
- H307 SIOモデル
- H308 HSI設定レジスタ
- H309 入力ポートモデル
- H310 HIP設定レジスタ
- H311 出力ポートモデル
- H313 HKD出力レジスタ

【図6】

機能関数選択テーブル3の例

機能関数	引数の条件
FETCH	(1000H ≤ ADRS < 2000H) OR KEYIN
IORD	(ADRS = READTIME) & I/O & RD
IOWR	(ADRS = PORTC) & D* & I/O & WR
CALL	(ADRS = KEYIN) & (0.5 ≤ TIME ≤ 1.5H)

【図7】

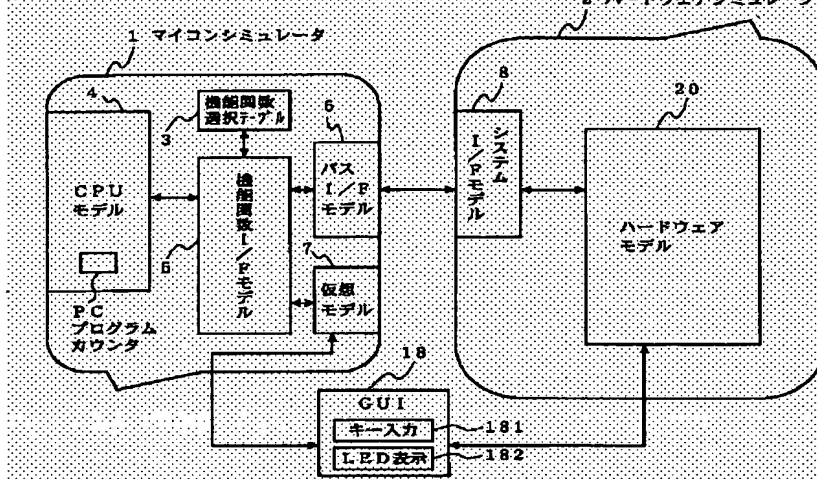
仮想関数変換テーブルとその戻り値

変換元	変換後	戻り値
FETCH (m)	MEMm (mは、1000H-1FFFFH)	命令コード
IORD (READTIME)	R_TM	時刻情報
IOWR (PORTC, 00)	LMP00	完了コード
IOWR (PORTC, 01)	LMP01	“
CALL (KEYIN)	K_IN	キーコード

【図1】

【図5】

本システムシミュレータ（第1実施例）の機能ブロック図



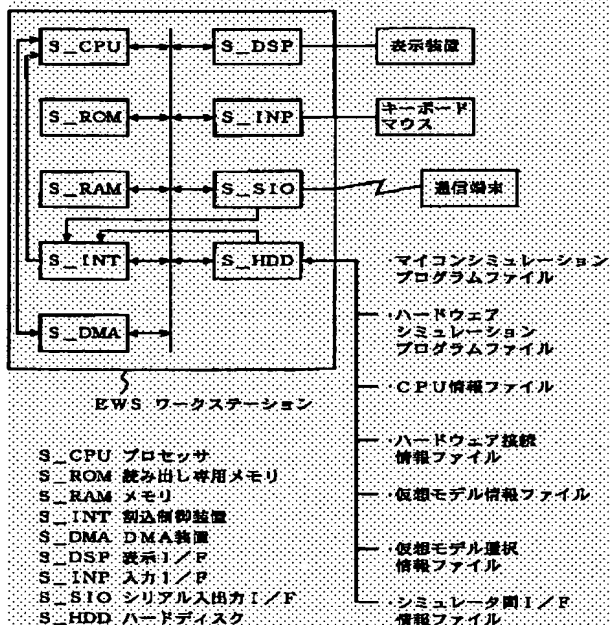
仮想プログラムメモリモデルV7.1内のテーブル例

シンボル	戻り値
MEM1001	A201
MEM1002	C801
MEM1003	B204
MEM1004	7406
MEM1005	2002
MEM1006	5AB5
MEM1007	B310
MEM1008	740A
MEM1009	2006
MEM100A	C701
MEM100B	5A03
MEM100C	B2EC
MEM100D	740F
MEM100E	200B
MEM100F	C700

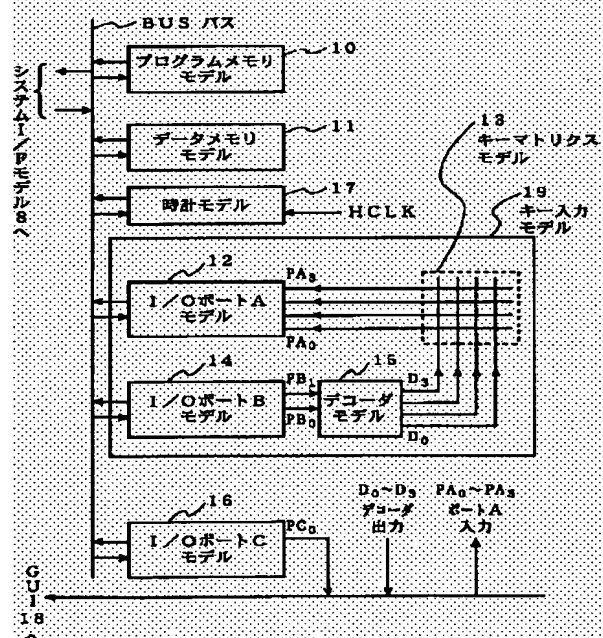
【図2】

【図3】

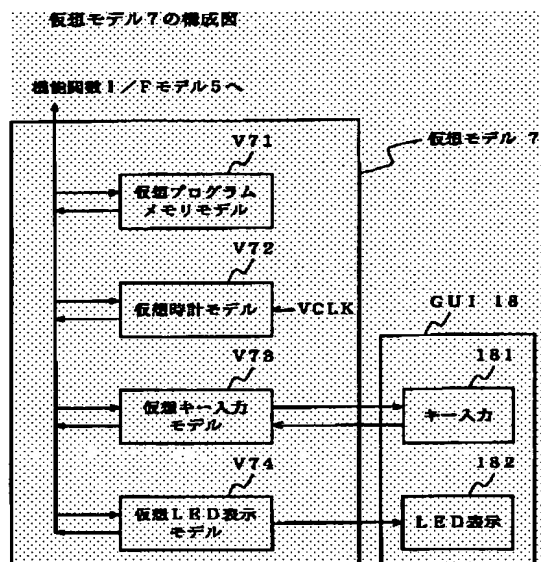
シミュレーション装置のブロック図



ハードウェアモデル20の構成例



【図4】



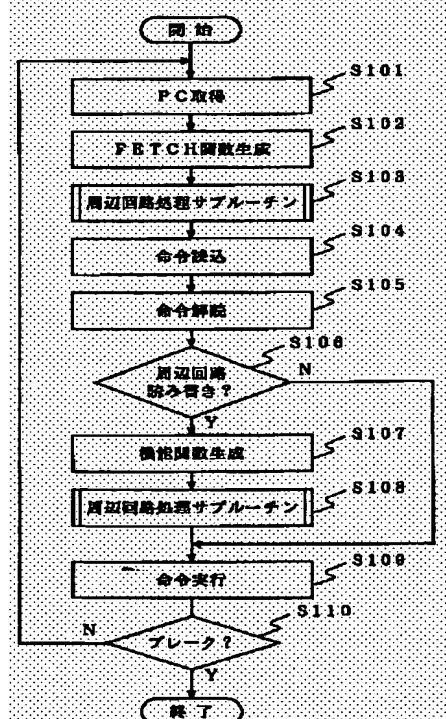
【図8】

ターゲット・プログラムリスト例

アドレス	命令コード	ニーモニック表示
1001	A201	MOV PORTB, #01H
1002	C501	MOV A, PORTA
1003	B204	SUB A, #04H
1004	7408	BZ 1008H
1005	2002	BR 1002H
1006	5AB5	MOV A, READTIME
1007	B310	SUB A, #10H
1008	740A	BZ 100AH
1009	2008	BR 1008H
100A	C701	MOV PORTC, #01H
100B	5A03	CALL KEYIN
100C	B2EC	SUB A, #ECH
100D	740F	BZ 100FH
100E	200B	BR 100BH
100F	C700	MOV PORTC, #00H
.....
3001	A201	MOV PORTB, #01H
3002	C501	MOV A, PORTA
.....
800A	7408	BZ 1300CH
800B	2002	BR 13002H
.....
301F	C8C9	RET

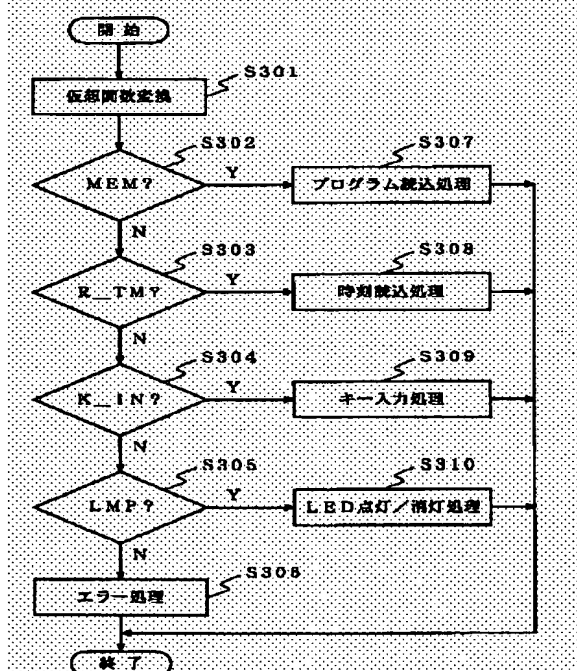
【図9】

CPUモデルシミュレーションのフローチャート

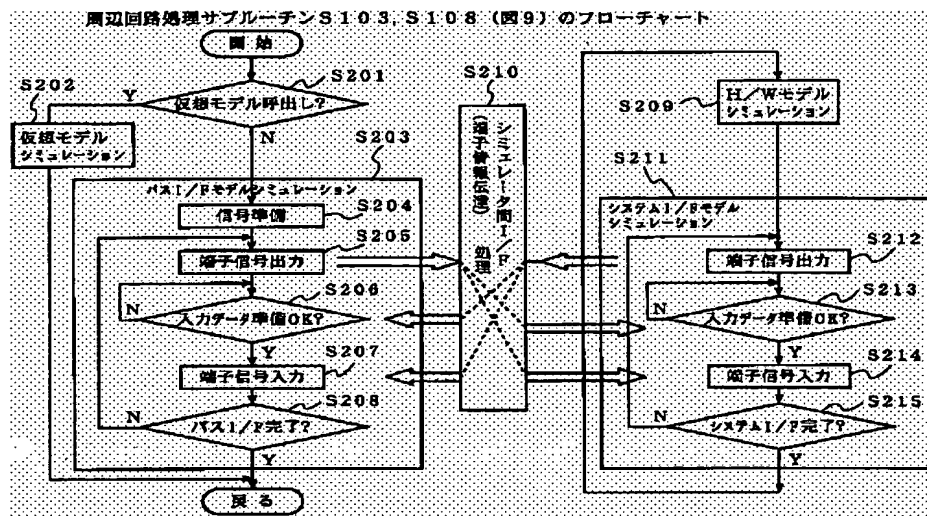


【図11】

仮想モデルシミュレーションS202 (図10) のフローチャート

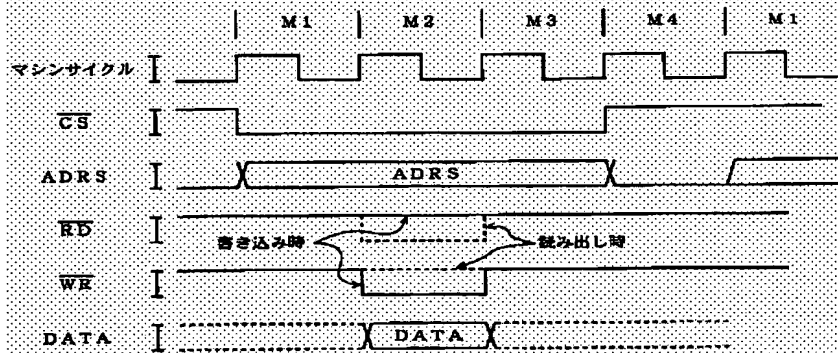


【図10】



【図12】

タイミングチャート (バス1/Fモデル6がハードウェアシミュレータ2にデータを読み書きするとき)

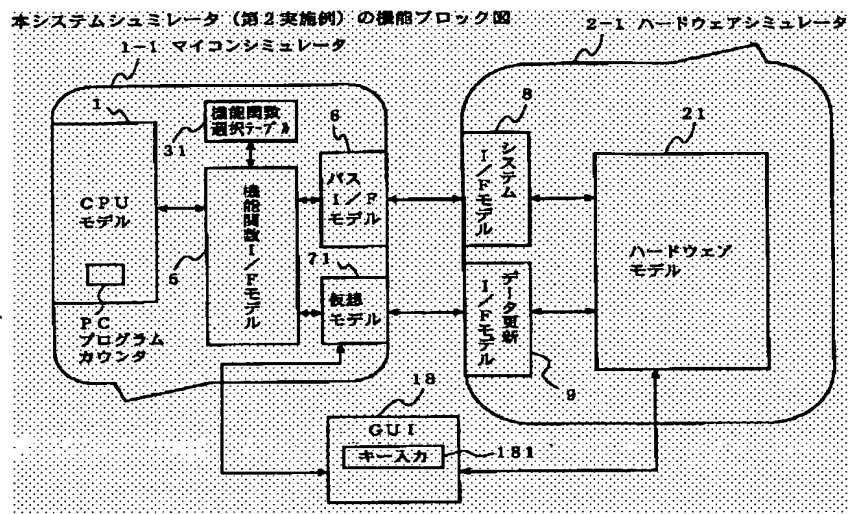


【図16】

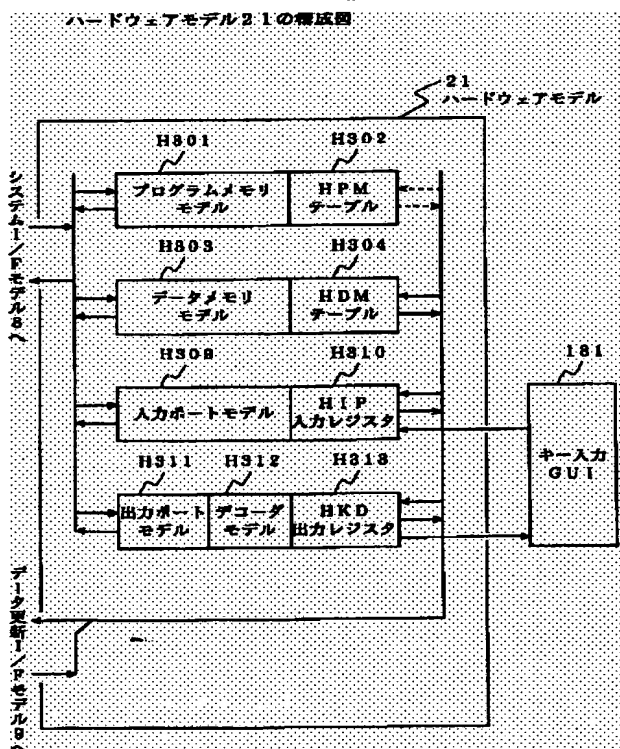
機能関数選択テーブル3-1の例

機能関数	引数の条件
FETCH	(1000H ≤ ADRS < 2000H)
IOWR	(ADRS = PORTB) & D* & I/O & WR

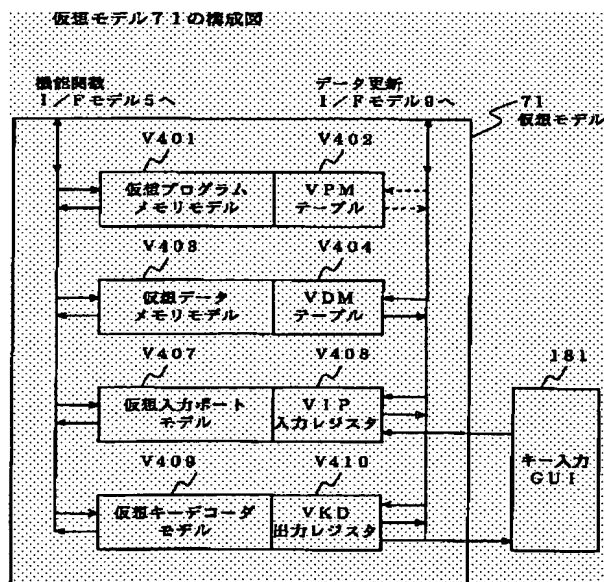
【図13】



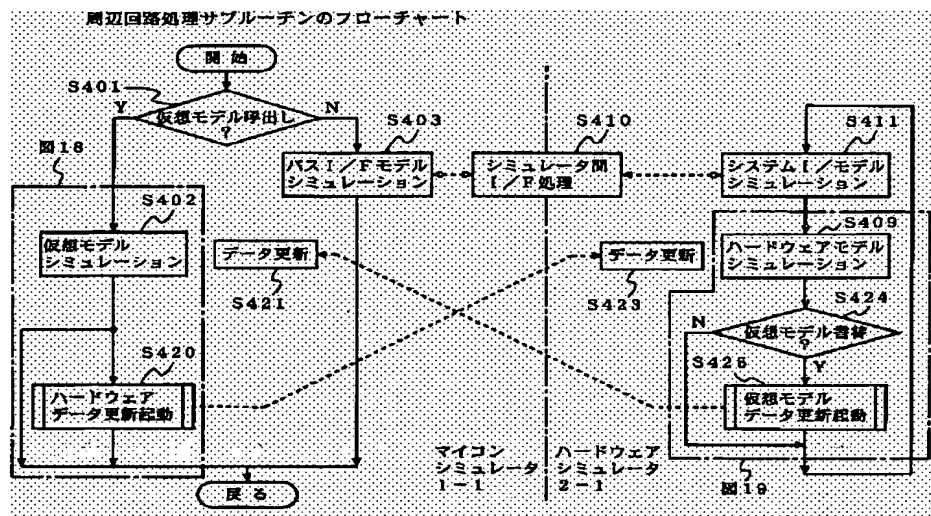
【図14】



【図15】



【図17】



【図18】

【図19】

仮想モデルシミュレーションS402 (図17) のフローチャート

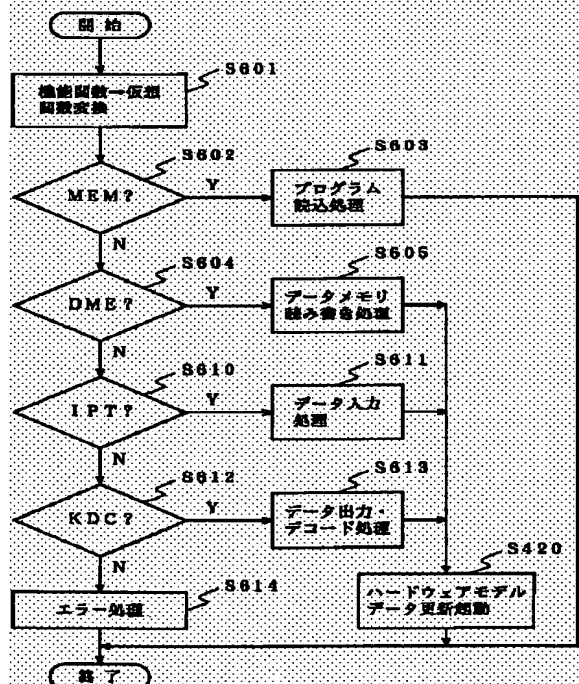
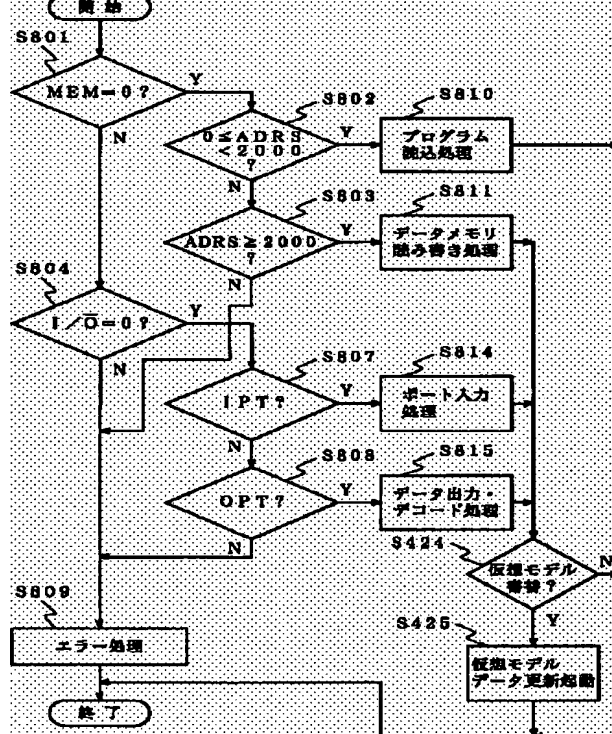
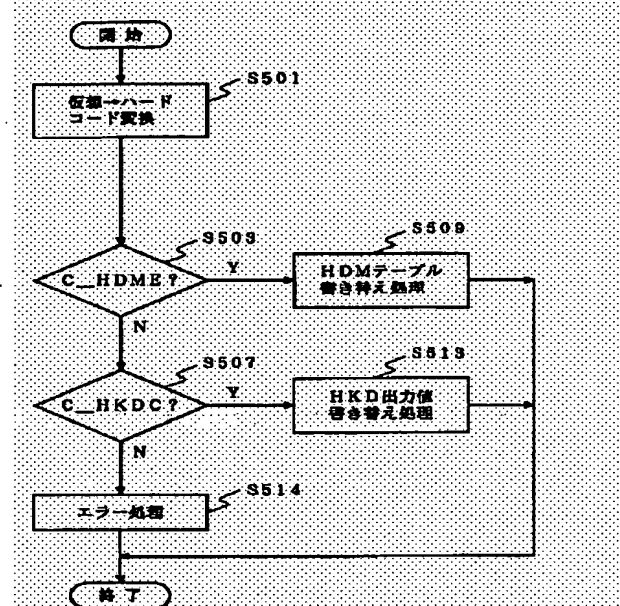


図17の一部詳細フローチャート



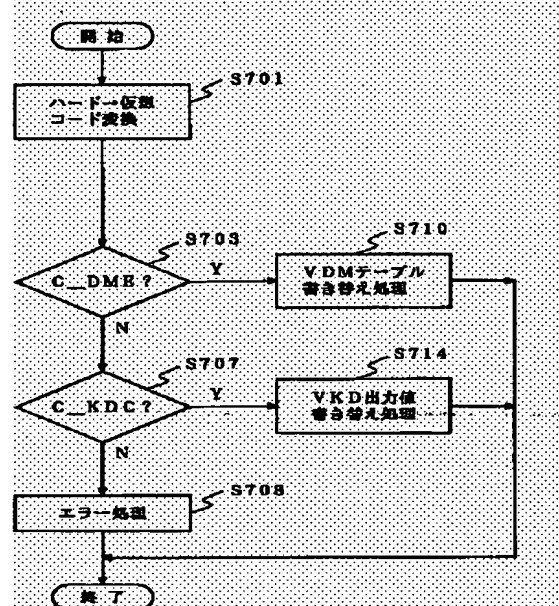
【図20】

ハードウェアモデルデータ更新S423(図17)の詳細フローチャート

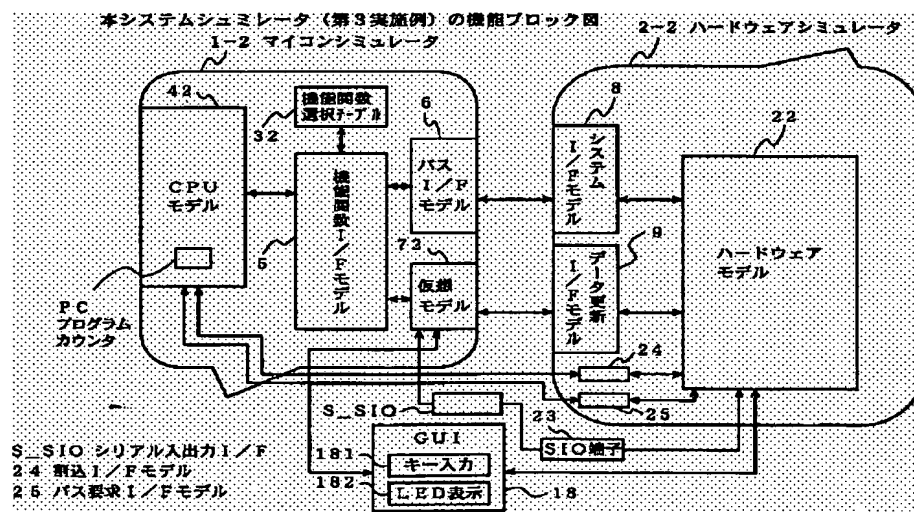


【図21】

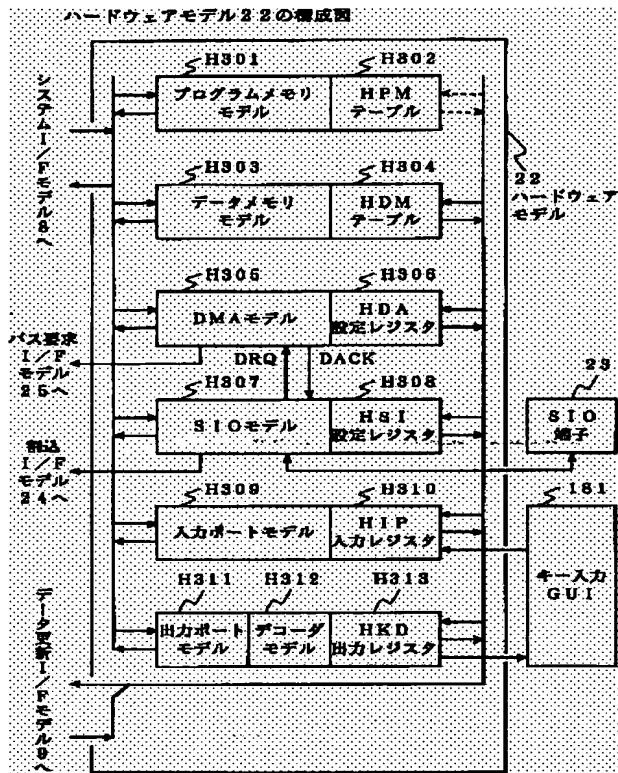
仮想モデルデータ更新S421(図17)の詳細フローチャート



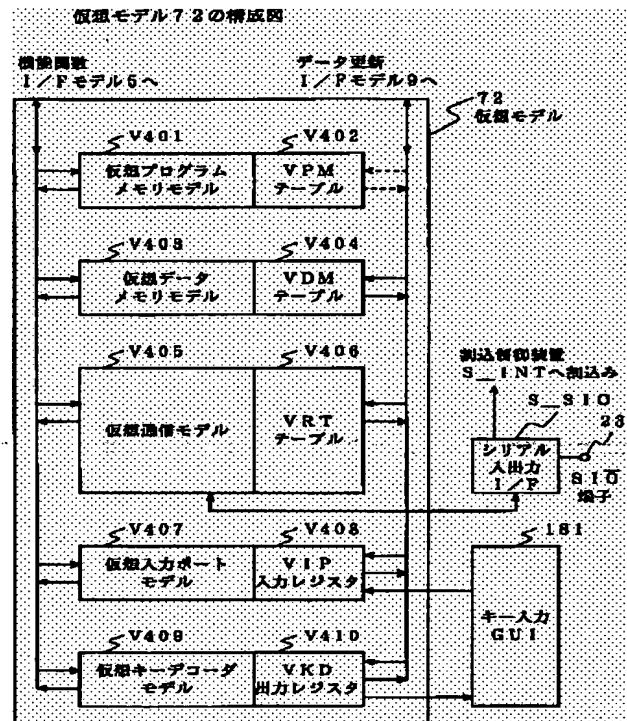
【図22】



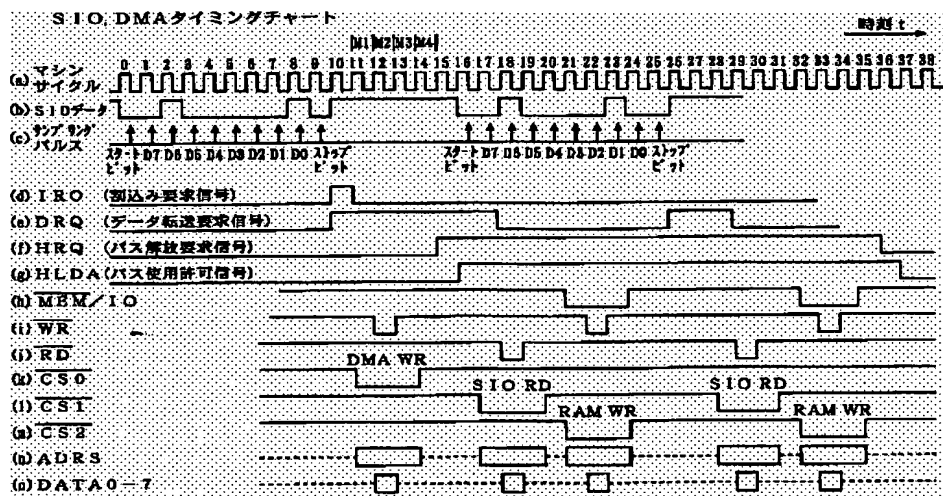
【図23】



【図24】



【図30】



【図25】

機能関数選択テーブル3.2の例

(A) 【ハードウェアモデルでSIO、DMA転送を行う場合】

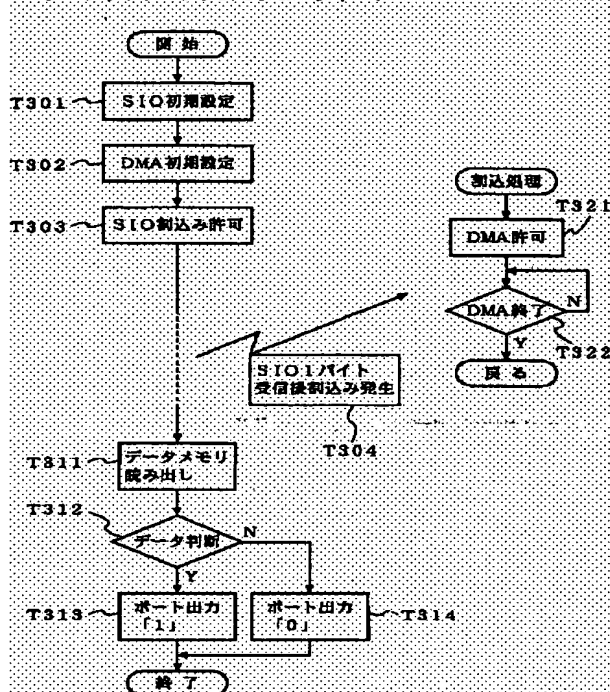
選択可能な関数	引数の条件	フラグ
FETCH	$(1000H \leq ADRS < 2000H) \& MEM \& RD$	1
MEM	$ADRS \geq 2000H \& MEM \& (RD \text{ or } WR)$	1
IORD	$(ADRS = PORTA) \& I/O \& RD$ $(ADRS = PORTB) \& I/O \& RD$ $(ADRS = DMA) \& I/O \& RD$ $(ADRS = SIO) \& I/O \& RD$	0 0 0 0
IOWR	$(ADRS = PORTA) \& D * I/O \& WR$ $(ADRS = PORTB) \& D * I/O \& WR$ $(ADRS = DMA) \& D * I/O \& WR$ $(ADRS = SIO) \& D * I/O \& WR$	0 0 0 0

(B) 【仮想モデルでSIO、DMA転送を行う場合】

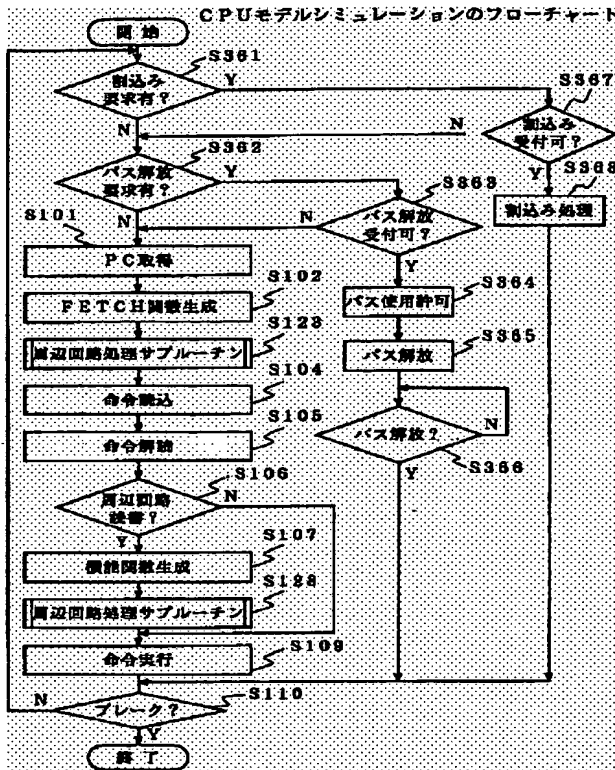
選択可能な関数	引数の条件	フラグ
FETCH	$(1000H \leq ADRS < 2000H) \& MEM \& RD$	1
MEM	$ADRS \geq 2000H \& MEM \& (RD \text{ or } WR)$	0
IORD	$(ADRS = PORTA) \& I/O \& RD$ $(ADRS = PORTB) \& I/O \& RD$ $(ADRS = DMA) \& I/O \& RD$ $(ADRS = SIO) \& I/O \& RD$	0 0 1 1
IOWR	$(ADRS = PORTA) \& D * I/O \& WR$ $(ADRS = PORTB) \& D * I/O \& WR$ $(ADRS = DMA) \& D * I/O \& WR$ $(ADRS = SIO) \& D * I/O \& WR$	0 0 1 1

【図26】

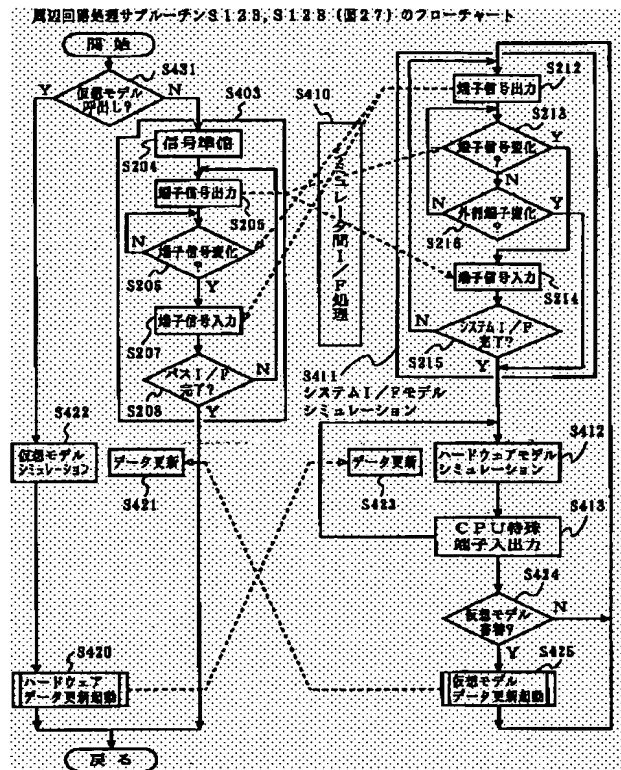
ターゲットプログラムのフローチャート



【図27】



【図28】



【図29】

